Modèles sémantiques et raisonnements réactif et narratif, pour la gestion du contexte en intelligence ambiante et en robotique ubiquitaire

Lyazid Sabri

UNIVERSITÉ PARIS-EST

École Doctorale Mathématiques et Sciences et Technologies de l'Information et de la Communication (MSTIC)

Spécialité : Signal, Image et Automatique

T H È S E

pour obtenir le titre de

Docteur en Sciences de l'université Paris-Est

Présentée par

Lyazid SABRI

Modèles sémantiques, raisonnements réactifs et narratifs, pour la gestion du contexte en intelligence ambiante et en robotique ubiquitaire

soutenue publiquement le 1er Juillet 2013

Jury :

Président :	Raja CHATILA	-	ISIR-Université Pierre et Marie Curie
Rapporteurs :	Rachid ALAMI	-	LAAS-CNRS
	François BREMOND	-	INRIA-STARS
Examinateurs :	Yacine AMIRAT	-	LISSI-Université Paris-Est Créteil
	Jean-Yves TIGLI	-	I3S-Université de Nice Sophia Antipolis
	Abdelghani CHIBANI	-	LISSI-Université Paris-Est Créteil
	Patrick GATELLIER	-	Invité, Thales Services
	Gian Piero ZARRI	-	Invité

Résumé : Avec l'apparition des paradigmes des systèmes ubiquitaires ou omniprésents, et de l'intelligence ambiante, on assiste à l'émergence d'un nouveau domaine de recherche visant à créer des environnements ou écosystèmes intelligents pouvant offrir une multitude de services permettant d'améliorer la qualité de vie, l'état physique et mental, et le bien-être social des usagers. Dans cette thèse, nous nous focalisons sur la problématique de la représentation sémantique des connaissances et du raisonnement dans le cadre des systèmes à intelligence ambiante et des robots ubiquitaires. Nous proposons deux modèles sémantiques permettant d'améliorer les fonctions cognitives de ces systèmes en termes de gestion du contexte. Au premier modèle, de type ontologique, sont associés un langage de règles et un raisonnement réactif pour la sensibilité au contexte. Pour prendre en compte le caractère dynamique du contexte et assurer une prise de décision cohérente, le mode de raisonnement retenu garantit deux propriétés essentielles : la décidabilité et la non-monotonie. Le deuxième modèle, également de type ontologique, complète le modèle précédent en termes d'expressivité pour la représentation de contextes non-triviaux et/ou liés au temps. Il s'appuie sur des relations n-aires et une représentation narrative des événements pour inférer des causalités entre événements et reconnaitre des contextes complexes non-observables à partir d'événements passés et courants. Les modèles proposés ont été mis en œuvre et validés sur la plateforme ubiquitaire d'expérimentation du LISSI à partir de trois scenarii d'assistance cognitive et de reconnaissance de contexte.

Mots clés : Intelligence ambiante, robotique ubiquitaire, modèles sémantiques, sensibilité au contexte, ontologies, raisonnement réactif, raisonnement narratif.

Abstract : With the appearance of the paradigms of ubiquitous systems and ambient intelligence, a new domain of research is emerging with the aim of creating intelligent environments and ecosystems, that can provide multiple services that can improve quality of life, the physical and mental status and the social wellness of the users. In this thesis, we address the problem of semantic knowledge representation and reasoning, in the context of ambient intelligent systems and ubiquitous robots. We propose two semantic models that improve the cognitive functions of these systems, in terms of context recognition, and context adaptation. The first one is an ontology-based model, which is associated with a rule language to model reactive reasoning process on contextual knowledge. To take into account the dynamicity of context and insure coherent decision-making, this process guarantees two essential properties : decidability and non-monotonic reasoning. The second model is also an ontology-based model that completes the previous model in terms of expressiveness for semantic representation of non-trivial contexts with temporal dimension It is based on n-ary relations and a narrative representation of events for inferring causalities between events, and therefore to build the chronological context of a situation as from past and current events. The proposed models have been implemented on the ubiquitous experimental platform of LISSI, and validated through three scenarios for cognitive assistance and context recognition.

Keywords : Ambient Intelligence, ubiquitous robotics, semantic modelling, context awareness, ontologies, reactive reasoning, narrative reasoning.

L'homme pense-t-il qu'on le laissera sans obligation à observer ?

Merci à celui que

Nul n'est égal à lui et

A enseigné à l'homme ce qu'il ne savait pas.

Hommage

Un éternel hommage à

SAKINA OUCHENE, LOUNES MATOUB et SALAH KABRI.

Vous avez allumé en nous le brasier de la liberté,
et le refus de la tyrannie.
Vous nous avez appris à faire face à l'affliction,
quand elle accourt vers nous.
De nous ravoir quand nous espérances sont anéanties, et même
enchaîner par la faim, que nous soyons rompu,
Nous refusons de s'accoter sur la patience, puisqu'il
n'est jamais d'espoir à guetter.

À ma sœur Taous. À ma sœur Saâdia, mes frères Rabah, Sedik et tous les enfants d'Elkoucha (Amirouche Amara, Mohend Seddik, Samir, Riadh, Badri, Abdelghani, Tahar, Nacer, Toufik, Karim, et bien d'autres) avec qui j'ai partagé la douleur et la joie.

Dédicaces

À mon épouse Saadia, pour le soutien moral inconditionnel et indubitable tout au long de cette thèse. À mes enfants **Mazir, Aksel et Tinhinnane**.

Remerciements

À Yacine Amirat, source d'inspiration et synonyme de sagesse.

À Gian Pierro Zarri, avec qui j'ai fait mes premiers pas vers la recherche scientifique. Oh ! Je suis parmi ceux qui ont eu la chance de te connaître.

À mon frère Abdelghani Chibani, sans lui, cette thèse n'aurait pu débuter et aboutir.

À l'honorable jury, Raja Chatila, Rachid Alami, Jean-Ye Tigli et François Brémond d'avoir accepté d'évaluer, d'examiner mes travaux ainsi que pour l'intérêt qu'ils ont manifesté à l'égard de ma contribution dans le domaine de l'intelligence ambiante et la robotique. Je tiens à remercier tous les membres de Thales. Je tiens à manifester tout mon respect et mon amitié à mes collègues et ami(e)s du LISSI.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction générale

La diversité des outils numériques utilisés dans nos activités quotidiennes, leur nombre, leur degré de sophistication sont en croissance perpétuelle. Ces dispositifs (terminaux de poche ou téléphones à écran tactile, tablettes numériques, etc. ou même robots domestiques) sont de plus en plus présents dans notre vie quotidienne. Ils constituent des outils d'assistance, de service, de communication et d'information, devenus à présent incontournables pour beaucoup d'entre nous. Ces évolutions technologiques permettent d'imaginer des systèmes intelligents offrant une multitude de fonctions accessibles, à n'importe quel moment, et à n'importe quel endroit, et selon une multitude de modes d'interactions et de média. On parle dans ce cas de systèmes ubiquitaires ou omniprésents, ou de systèmes intelligents ambiants. Avec l'apparition de ces paradigmes, on assiste aussi à l'émergence d'une nouvelle génération de robots de service appelés robots ubiquitaires. La particularité de tous ces systèmes réside dans leur capacité à adapter continuellement et automatiquement la même fonction ou service, à différents contextes et modes d'usage. À travers le paradigme de l'intelligence ambiante et de la robotique ubiquitaire, l'objectif est de créer des environnements ou des écosystèmes intelligents permettant d'améliorer l'environnement de vie des usagers. Ainsi, dans le cadre des applications de maintien à domicile des personnes âgées ou dépendantes, un système à intelligence ambiante (SIAM) peut offrir une multitude de services réactifs ou proactifs permettant d'améliorer la qualité de vie et l'état physique, mental, et le bien-être social des usagers. Ces services peuvent être de plusieurs types : Assistance à la mobilité, assistance cognitive, sécurité, surveillance médicale, maintien du lien social, etc.

Dans cette thèse, nous nous focalisons sur la problématique de la représentation sémantique des connaissances et du raisonnement dans le cadre des systèmes à intelligence ambiante et des robots ubiquitaires. Il s'agit d'étudier des solutions adaptées permettant d'améliorer les fonctions cognitives de ces systèmes en termes de gestion du contexte. Nous visons le développement de modèles sémantiques qui soient suffisamment expressifs et génériques, pour permettre une description symbolique à la fois pertinente et riche de ces systèmes. Ces modèles doivent permettre d'une part, la reconnaissance de contextes, en particulier, les contextes non-observables et non-triviaux et d'autre part, de mettre en œuvre des fonctions d'adaptation. Un contexte non-observable peut être vu comme le résultat de l'agrégation d'un ensemble de connaissances contextuelles observables. Nous considérons ici deux cas : L'exploitation des observations courantes et l'exploitation de l'historique des observations. Les modèles proposés doivent permettre par exemple de décrire les activités

de l'utilisateur allant des plus élémentaires (se lever/s'assoir, marcher, allumer le four, etc.) aux plus complexes telles que préparer un repas. Ces dernières peuvent être vues comme des séquences d'évènements associés à des activités élémentaires. Les objectifs de cette thèse concernent la proposition de deux modèles sémantiques répondant aux contraintes imposées par les systèmes à intelligence ambiante et les robots ubiquitaires, et leurs validations à travers la mise en œuvre de scénarii réalistes en utilisant la plateforme ubiquitaire développée au laboratoire LISSI.

Le premier modèle sémantique proposé vise la représentation explicite des entités physiques ou logiques (humain, robot, capteurs, actionneurs, services web, etc.), et des connaissances contextuelles associées à ces entités ; l'objectif étant d'exploiter ces connaissances dans les services sensibles au contexte. Pour satisfaire aux contraintes d'expressivité et d'extensibilité, nous proposons un modèle d'ontologie pour la description d'un environnement à intelligence ambiante s'appuyant sur le modèle proposé. Par ailleurs, il est nécessaire d'associer à ce modèle un langage de règles pour modéliser le processus de raisonnement réactif sur les connaissances contextuelles avec deux objectifs : La reconnaissance de contextes non-observables et la fourniture de services sensibles au contexte. Les contraintes imposées par les applications d'intelligence ambiante en termes de dynamicité du contexte et de cohérence dans la prise de décision, exigent des modes de raisonnement garantissant les propriétés essentielles suivantes : Décidabilité, non-monotonie du raisonnement. Par cette approche, l'objectif est de pallier aux problèmes rencontrés dans les ontologies du web sémantique qui s'appuient sur le raisonnement monotone, et qui sont par conséquent, inadaptées pour prendre en compte dynamiquement des changements de contexte ou pour inférer des contextes non-observables.

Le deuxième modèle vise, quant à lui, à compléter le modèle précédent en termes d'expressivité pour la représentation de contextes non-triviaux. Il s'agit ici de pallier aux inconvénients des approches ontologiques du Web sémantique en utilisant des relations n-aires pour la représentation des connaissances contextuelles. Sémantiquement, ce modèle doit permettre une représentation narrative des événements. Associé à des mécanismes d'inférence appropriés, ce modèle vise également à établir des relations sémantiques (causalité, but, etc.) implicites entre les évènements qui se produisent dans l'environnement. Plus précisément, il s'agit de modéliser les interdépendances entre contextes, et d'exprimer des connaissances de type : Qui est l'initiateur de l'événement/action ? dans quel contexte un événement est observé ou une action est exécutée ? quelle est l'entité qui a subit l'action ? etc. L'objectif est d'enrichir l'interprétation du contexte en vue d'assurer une meilleure adaptation à ce dernier.

Le troisième objectif de cette thèse concerne la mise en œuvre et la validation des modèles sémantiques proposés. L'évaluation de ces modèles ne doit pas se limiter à la mise en œuvre d'un environnement de simulation élémentaire avec des outils de tests unitaires, mais elle doit être réalisée dans un espace d'intelligence ambiante réel. Ce dernier doit être composé d'un ensemble de capteurs, actionneurs, robot, etc. L'environnement d'expérimentation doit reposer sur un intergiciel permettant de faciliter l'intégration de technologies matérielles et logicielles hétérogènes, et la mise en œuvre de services d'intelligence ambiante sensibles au contexte.

Le mémoire de cette thèse est composé de sept chapitres. Le présent chapitre constitue une introduction générale qui précise le contexte de l'étude, les contributions de la thèse et l'organisation du mémoire.

Dans le deuxième chapitre, nous introduisons tout d'abord les paradigmes de l'intelligence ambiante et des systèmes ubiquitaires, en considérant leurs intérêts pour l'émergence d'une nouvelle génération de robots de service appelés robots ubiquitaires. Dans la suite du chapitre, nous dressons un panorama des technologies actuelles et analysons les défis de la recherche concernant ces systèmes. Enfin, dans la dernière partie, nous esquissons les contours de nos travaux de recherche et synthétisons les objectifs de la thèse en termes d'amélioration des fonctions cognitives des systèmes à intelligence ambiante et des robots ubiquitaires pour la sensibilité au contexte.

Dans le troisième chapitre, nous présentons un état de l'art sur les formalismes de représentation symbolique des connaissances et les modèles de raisonnement associés. Nous faisons tout d'abord un rappel sur les concepts de la logique du premier ordre et du raisonnement logique qui représentent les briques de base communes à tous les systèmes de représentation et de raisonnement symbolique. Nous étudions ensuite les différents modèles de représentation de connaissances et de raisonnement à base d'ontologies, puis nous passons en revue les principaux travaux exploitant la représentation par ontologies dans le domaine de l'intelligence ambiante et de la robotique. La dernière partie du chapitre est consacrée à l'étude des principales approches de modélisation, de représentation temporelle et de manipulation des connaissances dynamiques.

Dans le quatrième chapitre, nous présentons le modèle sémantique proposé pour la représentation des connaissances contextuelles et le raisonnement réactif. Ce modèle, conçu pour répondre aux contraintes imposées par les applications d'intelligence ambiante, s'appuie sur deux langages : Le langage d'ontologie μConcept et le langage de règles *SmartRules*. Dans la première partie du chapitre, nous présentons les formalismes de ces deux langages. Dans la suite du chapitre, nous présentons l'ontologie *AmiOnt*, que nous avons proposée et développée à partir du langage μConcept. Cette ontologie permet de décrire tous les concepts nécessaires et tous les objets qui peuplent un environnement

d'intelligence ambiante (humains, robots, capteurs, actionneurs, robots, etc) et toute information, action ou événement pouvant être observée à partir de capteurs logiques ou physiques. Dans la dernière partie du chapitre, nous montrons comment la reconnaissance implicite de contextes et l'adaptation au contexte s'effectuent à partir d'un ensemble de règles *SmartRules* exprimées à partir de l'ontologie *AmiOnt*.

Le cinquième chapitre est consacré à la présentation du modèle de représentation de connaissances et de raisonnement NKRL (Narrative Knowledge Representation Language) pour la reconnaissance de contextes, à partir de l'historique des observations. Nous donnons tout d'abord les définitions utilisées tout au long de ce chapitre, puis nous présentons d'une part, les fondements de la modélisation des connaissances narratives à partir des ontologies HClass et Htemp et d'autre part, les mécanismes de raisonnement NKRL. Enfin, dans la dernière partie, nous développons une méthodologie d'annotation des événements élémentaires et dynamiques pour des applications en intelligence ambiante et en robotique ubiquitaire.

Le sixième chapitre est dédié à la mise en œuvre et à la validation expérimentale des modèles sémantiques et des raisonnements réactif et narratif proposés dans les chapitres 4 et 5 pour la gestion du contexte. Dans la première partie du chapitre, nous décrivons la plateforme ubiquitaire développée au laboratoire dans le cadre des ces travaux de thèse et des projets européens SEMBYSEM et WOO. Nous décrivons en particulier l'architecture logicielle d'implémentation pour la gestion du contexte. Dans la suite du chapitre, nous présentons l'implémentation de l'ontologie *AmiOnt* pour la mise en œuvre de scenarii d'assistance cognitive et de reconnaissance de contexte. Enfin, dans la dernière partie du chapitre, nous montrons, à travers un scénario de gestion de situations d'urgence, l'apport du raisonnement narratif pour la reconnaissance de contextes chronologiques.

Dans la dernière partie du mémoire, nous dressons un bilan des contributions et des perspectives de recherche découlant de ces travaux de thèse.

Chapitre 2

De l'intelligence ambiante à la robotique ubiquitaire : Principes, technologies et défis

Sommaire

2.1 Introduction

Dans ce chapitre, nous introduisons tout d'abord les paradigmes de l'intelligence ambiante et des systèmes ubiquitaires, en considérant leurs intérêts pour l'émergence d'une nouvelle génération de robots de service appelés robots ubiquitaires. Dans la suite du chapitre, nous dressons un panorama des technologies actuelles et analysons les défis de la recherche concernant ces systèmes. Enfin, dans la dernière partie, nous esquissons les contours de nos travaux de recherche et synthétisons les objectifs de la thèse en termes d'amélioration des fonctions cognitives des systèmes à intelligence ambiante et les robots ubiquitaires pour la reconnaissance du contexte et l'adaptation à ce dernier.

2.2 L'intelligence ambiante

La diversité des outils numériques utilisés dans nos activités quotidiennes, leur nombre, leur degré de sophistication sont en croissance perpétuelle. Ces

dispositifs (terminaux de poche ou téléphones à écran tactile, tablettes numériques, etc. ou même robots domestiques) sont de plus en plus présents dans notre vie quotidienne. Ils constituent des outils d'assistance, de service, de communication et d'information, devenus à présent incontournables pour beaucoup d'entre nous.

Ces évolutions technologiques permettent d'imaginer des systèmes intelligents offrant une multitude de fonctions accessibles, à n'importe quel moment, et à n'importe quel endroit, et selon une multitude de modes d'interactions et de média. On parle dans ce cas de systèmes ubiquitaires ou omniprésents. En effet, la particularité de ces systèmes réside dans leur capacité à adapter continuellement et automatiquement la même fonction ou service (par exemple le rappel de rendez-vous), à différents contextes et modes d'usage (affichage d'un message textuel sur le PC de travail de l'usager si ce dernier est sur son lieu de travail, annonce d'un message vocal depuis le téléphone portable de l'usager si ce dernier est dans sa voiture et qu'il est en train de conduire, etc.). Ces systèmes possèdent aussi la propriété de découvrir dynamiquement les objets immatériels (services d'informations divers) ou physiques (robots, capteurs, actionneurs, équipements multimédia, etc.) disponibles dans l'environnement, puis de les exploiter pour fournir des services aux usagers. Dans un tel environnement, le monde réel et le monde virtuel se mélangent pour transformer les dispositifs et équipements que nous utilisons dans notre vie quotidienne en objets communicants offrant pro-activement des services d'assistance à valeur ajoutée. Par exemple, grâce aux technologies d'identification par radio-étiquettes (tags) RFID et aux réseaux de communications sans fil à faible consommation électrique, des objets communicants représentant un capteur, un actionneur ou un objet physique quelconque, deviennent accessibles à large échelle selon le paradigme de l'Internet des objets. Cette déclinaison thématique de l'informatique ubiquitaire est appelée, intelligence ambiante ou Ambient Intelligence (AmI). Le terme intelligence ambiante a été introduit pour la première fois en 1998 par la société Philips dans le cadre d'une analyse prospective sur l'évolution de l'électronique grand public. En 2001, l'ISTAG (Information Societies Technology Advisory Group) a publié un document regroupant un ensemble de quatre scénarii, illustrant ce que pourrait être un monde à "Intelligence Ambiante" à l'horizon de 2010 [38]. L'objectif de ce travail était d'une part, d'alimenter sur un long terme la recherche et d'autre part, d'évaluer les recherches européennes dans ce domaine émergeant.

À travers ce paradigme, l'objectif est de créer des environnements ou des écosystèmes intelligents permettant d'améliorer l'environnement de vie des usagers. Ainsi, dans le cadre des applications de maintien à domicile des personnes âgées ou dépendantes, un système à intelligence ambiante (SIAM) peut offrir une multitude de services réactifs ou proactifs permettant d'améliorer la qualité de vie et l'état physique, mental, et le bien-être social des usagers. Ces services peuvent être de plusieurs types : Assistance à la mobilité, assistance cognitive, sécurité, surveillance médicale, maintien du lien social, etc. Il est ainsi possible d'imaginer une multitude

de services comme : Rappeler à l'usager de prendre ses médicaments, envoyer une alarme au corps hospitalier ou aux proches en cas d'accident (chute, intoxication, électrocution, brûlure, complication médicale, etc.), etc. D'autres exemples de services d'assistance au quotidien et agissant sur le long terme peuvent être envisagés pour permettre à l'usager de maintenir une bonne hygiène de vie, afin de prévenir ou réduire les effets d'éventuelles maladies chroniques (coaching d'activités physiques, stimulation cognitives, etc.). D'autres types de services peuvent être conçus comme ceux agissant sur l'environnement pour répondre aux préférences et aux besoins des usagers : Fermer/ouvrir les volets, allumer/éteindre la lumière, régler le confort ambiant, chercher un objet égaré, etc.

Il existe dans la littérature plusieurs définitions qui sont dans le fond assez similaires. Selon l'ISTAG [38], l'intelligence ambiante consiste à créer des environnements capables de prendre en compte les caractéristiques de chaque usager, de s'adapter et de répondre intelligemment à ses besoins spécifiques, d'agir de manière non intrusive et le plus souvent invisible, de permettre à l'usager d'accéder aux services de la façon la plus naturelle et intuitive possible, en exploitant la reconnaissance vocale, gestuelle ou la manipulation d'objets tangibles. Selon *Reignier* [112], l'intelligence ambiante est un paradigme résultant de l'intersection de l'informatique ubiquitaire et de l'intelligence artificielle. L'objectif consiste à exploiter les capacités de perception offertes par tous les capteurs afin d'analyser l'environnement, les utilisateurs et leurs activités, et de permettre au système de réagir en fonction du contexte. Une caractéristique importante de ce paradigme est la faculté d'analyse du contexte et l'adaptation dynamique aux changements de contexte. Ces notions seront développées dans la suite de ce chapitre.

2.3 Composition des environnements d'Intelligence Ambiante

Dans ce qui suit, nous dressons tout d'abord un panorama des technologies actuelles permettant de mettre en œuvre des systèmes à intelligence ambiante.

2.3.1 Artefacts intelligents

Un artefact intelligent correspond à n'importe quel objet physique fonctionnel de la vie quotidienne associant capteurs, unité de traitement, unité de communication et mémoire. Il est capable de percevoir son environnement, communiquer avec d'autres artefacts et éventuellement réagir selon une base de règles définie a priori. À travers sa capacité d'interaction direct avec un humain, un artefact intelligent peut assister une personne dans ses tâches quotidiennes en lui offrant des modes d'interaction intuitifs [166]. Contrairement aux dispositifs médicaux mobiles, les fonctionnalités médicales supplémentaires des artefacts intelligents ne sont généralement pas visibles de l'extérieur. Dans ce domaine, on peut citer

l'exemple de *Smart Pillow*, un oreiller intelligent développé par la société Philips. Ce système surveille les paramètres vitaux de l'utilisateur, tels que la température, la respiration, le pouls, et en cas d'urgence ou de maladie, avise le personnel médical [100]. Dans le domaine de l'assistance cognitive, on peut citer les travaux menés au laboratoire DOMUS de l'université de Sherbrooke-Canada. Le domicile est considéré comme une prothèse cognitive capable d'assister une personne ayant des déficits cognitifs (problèmes d'attention, de mémoire, de planification, etc.), par exemple en lui rappelant les tâches à réaliser, ou en l'aidant à gérer son temps ou à se préparer pour des rendez-vous [108]. Nous pouvons citer également les travaux de l'équipe *STARS* de l'*INRIA* sur l'analyse des comportements des personnes atteintes de la maladie d'Alzheimer [154].

Le concept d'artefacts intelligents a été étudié et développé dans des applications autres que les médicales. *Smart Sofa* est un canapé instrumenté qui a été développé par le Trinity College de Dublin-Irelande. Il permet d'identifier les personnes assises dessus et de fournir des services personnalisés basés sur ces informations [68]. Les ustensiles intelligents de cuisine sont des exemples d'artefacts mis en œuvre au Massachusetts Institute of Technology [77] : (i) une casserole, équipée d'une puce, qui indique si elle est trop chaude pour être manipulée ; (ii) une cuillère qui fournit des informations sur la température et la viscosité de la nourriture ; (iii) une bouilloire qui informe l'utilisateur du temps d'attente pour la préparation de son thé [31]. D'autres prototypes d'artefacts intelligents ont été développés dans la même logique comme la tasse de café qui communique le type de café et la température du liquide qu'elle contient ; ou bien encore, la nappe interactive qui permet de saisir une commande dans un restaurant [106].

2.3.2 Accessoires et Vêtements Intelligents

Concernant les technologies mobiles pour la santé et l'autonomie, deux grands courants de la recherche sont devenus prédominants au cours des dernières années : Les accessoires intelligents et les vêtements intelligents.

Concernant la première catégorie, l'exemple le plus marquant est le projet Google Glass, une paire de lunettes intelligente qui offre des services de communication et de navigation à l'usager mobile. *Starner et al.* [43] ont développé *Gesture Pendant*, un pendentif qui reconnaît des gestes prédéfinis de l'usager et exécute des actions de contrôle correspondantes. *Kikin-Gil et al.* ont développé des accessoires intelligents pour permettre la communication non-verbale au sein de petits groupes d'adolescents [61]. Les montres ou bracelets sont des accessoires populaires qui sont de plus en plus utilisées dans des applications de surveillance médicale. Plusieurs modèles de montres intelligentes sont déjà disponibles dans le commerce à l'image de *l'Actiwatch* de la société Cambridge Technology. Cette montre est équipée d'un accéléromètre miniature qui mesure l'activité physique de son porteur. La combinaison de ce capteur avec d'autres capteurs a permit d'étendre les fonctionnalités

de la montre à la surveillance des troubles du sommeil : Insomnie, humeur, dépense d'énergie, mouvements périodiques des membres pendant le sommeil [71]. Il existe d'autres exemples de montres-bracelets offrant d'autres types de fonctionnalités, telles que la détection de chute[1], le test de fibrose kystique [5], la mesure de glycémie [141], la surveillance de l'oxygénation du sang [147], l'envoi d'appels d'urgence [159].

Le concept de vêtement intelligent (wearable computing- informatique à porter) repose sur l'idée qui consiste à avoir des ordinateurs miniatures comme partie intégrante des vêtements qui nous habillent ou des accessoires que nous portons. Les vêtements intelligents sont des artefacts portables destinés à accompagner l'utilisateur dans ses déplacements. Dans cette catégorie, on peut citer le projet *Smart Shirt* développé à Georgia Institute of Technology [1], où différents types de capteurs ont été intégrés dans la conception d'une chemise intelligente, pour permettre le suivi de paramètres vitaux comme la fréquence cardiaque, l'électrocardiogramme (ECG), la respiration, la température, etc. [46] et [33]. Un autre exemple de système portable de surveillance de paramètres physiologiques est la veste intelligente *(Smart Vest)* développée par *Pandian et al.* [99]. Le système permet de surveiller des paramètres physiologiques tels que l'ECG, la pression artérielle, la température du corps et la fréquence cardiaque. Ces paramètres ainsi que la géo-localisation du porteur sont transmis aux stations de surveillance [98]. Certains systèmes sont déjà commercialisés tels que le gilet LifeShirt de la société VivoMetrics. Ce système permet d'effectuer de manière non-invasive une pléthysmographie respiratoire inductive pour surveiller des paramètres cardiorespiratoires (mesure des variations de volume du thorax et de l'abdomen dues aux mouvements respiratoires) [50]. *Actibelt*, est une ceinture intelligente qui est équipée d'un accéléromètre tridimensionnel intégré dans sa boucle. Ce capteur permet d'analyser l'activité physique du porteur sur une longue période [129]. On peut trouver dans la littérature d'autres exemples de vêtements intelligents : Protections actives de la hanche [132], chaussures intelligentes [58], genouillères intelligentes [95].

2.3.3 Implants intelligents

Les implants dentaires constituent probablement la forme la plus largement répandue des implants médicaux. Un système de télémétrie et des capteurs sont généralement intégrés à l'intérieur de la bouche du patient dans une dent artificielle ou une couronne. L'un des premiers prototypes d'implants dentaires a été développé dans le cadre du projet Européen Saliwell et était destiné aux patients souffrant de déficience de production de salive. Le prototype, basé sur un capteur de salive, surveille en permanence l'état de sécheresse de la bouche et restaure automatiquement la production de salive naturelle au moyen de l'électrostimulation [131]. Le système IntelliDrug est un autre exemple de prothèse dentaire qui comprend un dispositif automatisé d'injection de doses de médicaments [131]. Cet implant est

1. http://www.cleode.fr

destiné aux personnes souffrant de pharmacodépendance ou de maladies chroniques. Le projet Européen Healthy Aims vise l'élaboration d'implants médicaux incluant des capteurs et des microsystèmes à l'échelle nano-matériel : Implant rétinien pour restaurer la vision pour des patients atteints de certains types de cécité, implant cochléaire destiné à rétablir l'audition, implant pour la stimulation électrique fonctionnelle (FES) des muscles des membres inférieurs, capteur de glaucome, capteur de pression intracrânienne pour le diagnostic des malades souffrant d'hydrocéphalie, capteur de mouvement pour la surveillance de l'activité physio-électrique de l'organisme [2].

2.3.4 Identification

Parmi les systèmes embarqués capables d'identifier ou de suivre une personne ou un objet, on peut citer l'identification à base de radiofréquences appelée communément RFID (Radio Frequency IDentification). Cette technologie, actuellement mature, connaît une véritable explosion, en particulier grâce aux efforts multiples et continus de miniaturisation des circuits intégrés. Son principe consiste en des radio-étiquettes (tags) RFID qui sont capables d'émettre un signal contenant un identifiant en réponse à une requête envoyée par un lecteur. On distingue deux types d'étiquettes : Les étiquettes passives et les étiquettes actives. Les étiquettes passives (sans batterie ou pile) sont alimentées par le champ électromagnétique généré par le lecteur à l'émission d'une requête. Les étiquettes actives, quant à elles, sont alimentées par une batterie ou une pile. Outre l'identification, les étiquettes actives peuvent potentiellement êtres utilisées comme télémètres basés sur la mesure de la puissance du signal Radiofréquence. On dénombre quatre bandes principales de fréquences dédiées pour les applications de la technologie RFID : Les basses fréquences, aux alentours de 125 kHz, les hautes fréquences à 13,56 MHz, la bande UHF, entre 800 et 900 MHz et enfin les hyperfréquences avec des fréquences à 2,45 GHz et 5,8 GHz.

La figure 2.1 représente le robot EL-E, un prototype de robot d'assistance développé au Georgia Institute of Technology. L'identification RFID permet au robot de percevoir et de comprendre sémantiquement son environnement, d'interagir avec des utilisateurs et de manipuler des objets identifiés par leurs labels [36]. Des antennes de longue portée sont utilisées pour détecter la présence d'un objet ou d'une personne dans un espace (chambre, cuisine, etc.). Les antennes de courte portée permettent au robot de détecter la présence d'un objet à proximité de sa main.

La figure 2.2 illustre l'instrumentation d'un appartement dans le cadre du projet PEIS Ecology mené par l'université D'Orebro-Suède. Le système utilise une grille de tags RFID passifs sous le parquet pour guider le robot Centribot. Ce dernier est équipé d'un lecteur de tags RFID, dont l'antenne s'étend depuis la base.

2. http://www.healthyaims.org/

FIGURE 2.1 – Robot EL-E, équipé d'un lecteur d'étiquettes RFID UHF, remettant une boite de médicaments munie d'une étiquette RFID à une personne portant un bracelet équipé d'une étiquette (tag) RFID [36].

a) Grille d'étiquettes RFID b) Robot Centribot

FIGURE 2.2 – Instrumentation du sol d'un appartement par des étiquettes RFID passives [115].

2.3.5 Localisation

Dans cette partie, nous nous limitons à l'étude des principaux systèmes de localisation indoor. Dans cette catégorie, on distingue deux types de systèmes [128] :

- Les systèmes WPS (Wifi Positioning Systems) : La localisation par ondes WIFI peut s'effectuer selon différents principes ; le plus simple consiste en une triangulation entre des bornes WIFI. L'un des plus performants en termes de précision utilise la méthode de cartographie ou d'empreinte radio (radio fingerprinting). Parmi les systèmes utilisant ce principe, on peut citer le Système RTLS (Real Time Location

System) de la société Ekahau qui offre une précision de l'ordre de quelques mètres [3]. La puissance du signal reçu ou RSS (Received Signal Strength) est le paramètre le plus souvent utilisé dans les méthodes de localisation WPS (triangulation, cartographie). Cependant, cette technique fait l'hypothèse que le modèle d'atténuation des lieux (obstacles, murs, etc.) soit bien connu, ou calibré par apprentissage.

- Les systèmes de localisation basés sur les technologies sans fil courte distance. Parmi ces technologies, on peut citer : Bluetooth, Infrarouge, Zigbee, ultra-large bande (Ultra Wideband ou UWB), etc. Les méthodes de localisation utilisées dans les systèmes WPS sont également applicables pour ces systèmes.

Active Badge, est un système de localisation utilisant les ondes infrarouges (IR). Une étiquette portée par une personne émet un signal IR toutes les 10 secondes. Des capteurs placés à des endroits spécifiques du site captent ces signaux et les envoient à un calculateur en réseau qui estime la position de l'étiquette [150]. Active Badge possède une précision relativement faible de l'ordre de plusieurs mètres. Il est également très sensible à la lumière fluorescente et à la lumière du soleil.

Cricket, un système conçu à l'origine par le MIT, utilise une combinaison des ondes RF (Radio fréquence) et des ondes ultrasons, figure 2.3. Des balises "Cricket beacons" déployés sur le site envoient des ondes RF et ultrasonores au récepteur "Cricket listener" attaché à l'objet mobile. La position de ce dernier est ensuite estimée à partir de la mesure de la différence entre le temps de propagation des ondes RF et des ondes ultrasonores. Ce paramètre est appelé, différence de temps d'arrivée ou TDoA (Time Difference of Arrival) [109]. La précision de localisation est de l'ordre de 10 cm.

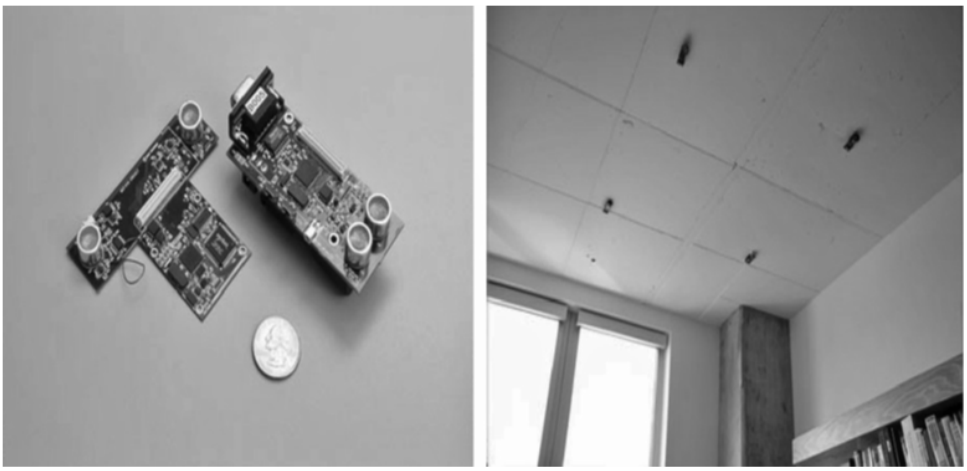

FIGURE 2.3 – Système de localisation Cricket.

3. http://www.ekahau.com/products/real-time-location-system/overview.html

Le système Ubisense, utilise, quant à lui, la technologie UWB,[4] figure 2.4. La localisation d'une balise active (appelée Ubitag) portée par l'objet mobile s'effectue selon le principe de la triangulation à partir d'une collection de capteurs en réseau (appelés Ubisensors) déployés dans une cellule rectangulaire. Chaque Ubitag intègre d'une part, une radio UWB (6-8 GHz) pour l'envoie de signaux UWB impulsionnels de très courte durée qui sont captés par les Ubisensors et d'autre part, une radio conventionnelle RF (2,4 GHz) pour la coordination de l'envoi de signaux UWB. Le système Ubisense utilise les paramètres TDoA (la différence de temps d'arrivée) et AOA (l'angle d'arrivée correspondant à la direction du signal -Angle of Arrival) pour calculer par triangulation la position de la balise Ubitag. Ainsi, au moins deux capteurs Ubisensors sont nécessaires pour calculer la position 3D d'une balise Ubitag. La localisation dans un appartement ou bâtiment nécessite que le site soit divisé en cellules de forme rectangulaire (chambre, séjour, bureau, etc.) nécessitant chacun 3 capteurs. Le système assure une précision relativement élevée (de l'ordre de 15 centimètres) mais reste cependant très onéreux et nécessite une infrastructure avec un câblage spécifique pour la mise en réseau des capteurs.

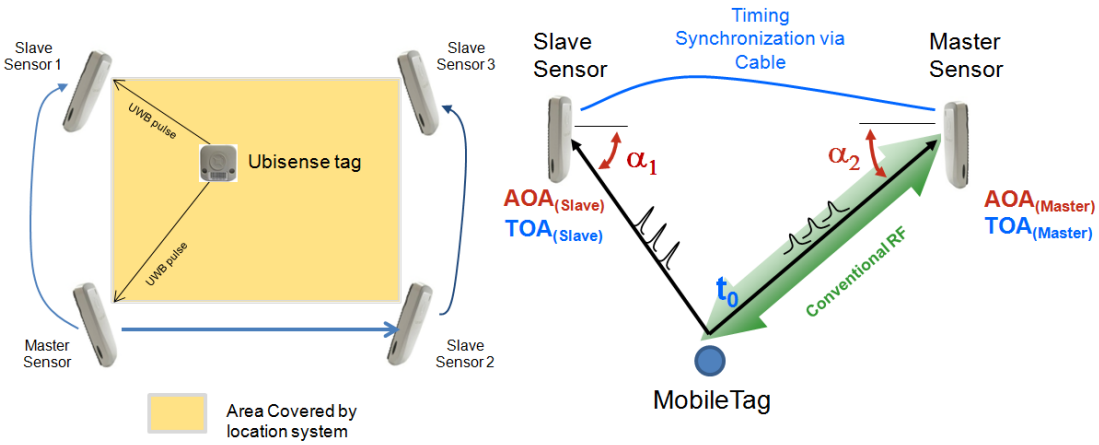

FIGURE 2.4 – Système de localisation Ubisens.

2.3.6 Robots de services

L'utilisation de robots de services pour l'assistance aux personnes dans l'exécution de leurs tâches ou activités quotidiennes est toujours en phase d'expérimentation et de recherche de modèles économiques viables. Dans ce qui suit, nous nous limiterons aux deux catégories suivantes : Les robots d'assistance à la mobilité ou au déplacement et les robots d'assistance domestique.

4. http://www.ubisense.net/en/

2.3.6.1 Robots d'assistance à la mobilité

Ces robots sont en général conçus pour compenser les déficiences motrices de leurs utilisateurs et aider ces derniers dans leurs activités physiques quotidiennes comme se lever/s'asseoir, marcher, monter les escaliers, etc. Dans cette catégorie de robots, on trouve les robots fauteuils, les robots cannes, les robots déambulateurs et les robots exosquelettes. La figure 2.5 illustre quelques prototypes de robots : (a) le concept de fauteuil roulant robotisé CARRIER développé par l'Université des arts appliqués de l'Industrial Design Studio 2 Esslinger en Autriche. Ce robot fait également office de verticalisateur ; (b) le fauteuil roulant robotisé Sharioto de l'Université de Louvain [34] ; (c) La cane intelligente iCane développée à l'université de Nagoya [37] ; (d) le déambulateur RobuWalker développé par la société Robosoft et l'ISIR pour l'aide à la verticalisation et au déplacement [5] ; (e) l'exosquelette Hal de la société Cyderdyne [6] ; (f) l'exosquelette EiCOSI, pour l'assistance aux mouvements de l'articulation du genou, développé au laboratoire LISSI [89]. Les exosquelettes ou orthèses sont des dispositifs mécatroniques que l'on qualifie de robots portables. Ils sont utilisés dans le but d'augmenter, d'assister ou de restaurer les mouvements des personnes dépendantes.

FIGURE 2.5 – Robots d'assistance à la mobilité : (a) CARRIER (b) Sharioto (c) iCane (d) RobuWalker (e) Hal (f)EiCOSI.

5. http://www.robosoft.com
6. http://www.cyberdyne.jp/english/customer/index.html

2.3.6.2 Les robots d'assistance domestique

Ce domaine s'est développé ces dernières années pour le grand public, avec principalement l'apparition sur le marché de robots de service mono-tâche, à autonomie limitée et à prix réduit, comme par exemple, le robot de nettoyage Roomba de iRobot. Ces robots grand public nécessitent encore des évolutions pour s'adapter aux capacités fonctionnelles des personnes âgées ou dépendantes.

Dans la catégorie des robots d'assistance domestique qu'on appelle aussi robots personnels, on peut également citer les robots d'assistance relationnelle qui regroupent les robots compagnons et les robots pour l'éveil sensoriel.

Les robots compagnons ont une fonction d'assistance cognitive auprès de personnes isolées et fragilisées et peuvent communiquer avec la personne en tant qu'objets mobiles et communicants. Les robots compagnons sont en général constitués d'une base mobile avec une tête. Ils sont munis de capteurs, de reconnaissance et de synthèse vocale, d'interfaces de connexion à Internet et de fonctions de navigation autonome. Les fonctions principales pouvant être prises en charge par un robot compagnon concernent essentiellement l'assistance cognitive, allant des aide-mémoire aux stimulations (exercices physiques et intellectuels), ainsi que les fonctions basiques de communication comme le courrier électronique ou l'interaction social (accès aux réseaux sociaux, famille, amis, etc.). La figure 2.6 illustre quelques exemples de robots personnels : Kompai de Robosoft, Ava d'iRobot, PR2 de Willow Garage.

FIGURE 2.6 – Robots personnels : (a) Kompai (b) Ava (c) PR2.

Les robots d'éveil sensoriel sont généralement conçus pour favoriser l'interaction et la communication par le contact direct avec la personne. Ils sont destinés aux personnes souffrant de troubles cognitifs tels que l'autisme, les maladies

d'Alzheimer, etc. Ils se présentent sous la forme d'animaux de compagnie robotisés à l'image du robot chien Aibo de Sony. Ces robots sont équipés de capteurs leur permettant de réagir au toucher et à la voix et peuvent émettre des sons. Ils sont dotés d'une intelligence leur permettant essentiellement une communication et une interaction affective, personnalisée avec l'individu.

Le robot PR2 de Willow Garage est sans doute le robot personnel le plus abouti. Il dispose d'une base mobile à roues et de deux bras qui possèdent chacun 7 degrés de liberté. Ce robot reste néanmoins un outil d'expérimentation pour la recherche. La robotique de service multitâche pour des applications domestiques est toujours au stade de l'expérimentation et de la recherche, comme en témoignent les nombreux travaux, notamment sur les robots humanoïdes. A ce jour, il n'existe pas encore de produits commercialisés dans le domaine grand public.

2.3.7 Vers la robotique ubiquitaire

Avec l'apparition des paradigmes de l'informatique ubiquitaire, de l'intelligence ambiante, et la disponibilité des technologies de communications sans fil (réseaux de capteurs, réseaux mobiles, smart phones, etc.), des technologies logicielles (Web services, middleware), d'identification ($RFID$, biométrie), etc., on assiste à l'émergence d'une nouvelle génération de robots de service appelés robots ubiquitaires. La robotique ubiquitaire ou la robotique en réseau constitue un domaine de recherche très prometteur de la robotique [62] [124]. Les enjeux autour de cette nouvelle forme de la robotique sont considérables avec un potentiel fort et des perspectives de croissance prometteuses, notamment sur le marché de la robotique de service. Des acteurs de l'industrie informatique de tout premier plan comme Google, Microsoft, Intel, etc. s'intéressent de plus en plus à ce marché.

L'idée sous-jacente consiste à intégrer ces robots dans des espaces intelligents à petite ou large échelle (maisons intelligentes, bâtiments, espaces urbains), afin d'offrir, en tout lieu, à tout instant et de manière transparente des services à valeur ajoutée : Assistance cognitive, sécurité, confort, divertissement, etc. Nous définissons un service comme une assistance (ou commodité matérielle, physique ou immatérielle) fournie à un (ou des) êtres vivant(s) et notamment à des personnes. La notion de service est apparue récemment dans les sciences informatiques. Dans ce cadre, un service offre une fonction utilisable par un tiers. Ce tiers peut être un ou plusieurs êtres vivants ou un autre service. La notion de service telle qu'elle est conçue dans les systèmes d'information distribués peut ainsi être perçue de la même manière qu'en robotique ubiquitaire. Le modèle d'interaction entre le robot et son environnement est ainsi totalement repensé puisqu'il ne s'appuie plus sur un schéma pré établi ; l'environnement d'évolution étant considéré comme ouvert, partiellement connu et dynamique. Un robot ubiquitaire doit être interopérable avec des objets ou des entités (capteurs, actionneurs, robots, artefacts, etc.) qu'il découvre dans son environnement plutôt que d'être pré

programmé statiquement pour cet environnement. Il doit être capable non seulement d'utiliser ses propres objets (capteurs et actionneurs) mais aussi d'interagir avec d'autres objets de l'environnement et d'adapter dynamiquement ses services face à des changements de contextes. On parle dans ce cas de sensibilité au contexte.

Le chevauchement qui existe actuellement entre la robotique ubiquitaire et l'intelligence ambiante fait que leur utilisation conjointe constitue un choix synergétique pour créer des espaces physiques et numériques riches et offrant une multitude de services intelligents réactifs et/ou proactifs, visant à améliorer nos cadres de vie en tout lieu (maison, travail, etc.) et à tout instant. L'intelligence ambiante, d'une manière générale, et la robotique ubiquitaire, en particulier, constituent un domaine de recherche en plein essor qui vise à créer des écosystèmes exploitant des objets communicants (capteurs, actionneurs, terminaux numériques, artefacts intelligents, robots d'assistance, etc.) d'un environnement connecté, pour créer des services et des espaces (web et physiques) intelligents selon la vision du Web des Objets (Web of Things).

L'apparition récente du concept du cloud computing (informatique en nuage) va dans le même sens et va permettre l'émergence d'une nouvelle génération de robots ubiquitaires capables d'enrichir leurs capacités cognitives et partager leurs connaissances en se connectant à des infrastructures cloud [60] [48] [65]. Ces robots ouvrent ainsi la voie à un nombre illimité d'applications telles que les compagnons physiques et virtuels pour aider les personnes dans leur vie quotidienne. Ces robots seront capables de travailler aux côtés des personnes et de coopérer avec ces dernières en partageant le même environnement ; ils agiront en tant que gardes physiques et/ou virtuels autonomes pour protéger les personnes, surveiller leur sécurité et leur venir en aide dans les espaces intérieurs et extérieurs.

Selon cette nouvelle vision de la robotique, un robot ubiquitaire sera susceptible aussi de servir d'incarnation pour l'interaction avec les humains. Ainsi, les travaux de recherche autour du concept de l'affective computing (l'informatique affective) explorent déjà l'usage des modèles articulés des visages. Couplés à une interaction vocale, de tels dispositifs pourront créer une illusion forte d'intelligence, quasi humaine. Une application naturelle de ces technologies consisterait à associer à un système à intelligence ambiante l'apparence d'un visage humain ou à l'incarner à travers un ou plusieurs objets de la vie quotidienne.

2.4 Défis de l'intelligence ambiante et de la robotique ubiquitaire

Tout système utilisant les paradigmes de la robotique ubiquitaire et de l'intelligence ambiante doit posséder les caractéristiques essentielles suivantes : L'interopérabilité pour assurer l'omniprésence et la continuité du service pendant

la mobilité, l'auto-localisation, l'auto-exploration de l'environnement physique, la sensibilité au contexte et à l'intention des usagers, l'interaction multimodale et naturelle avec les autres systèmes, la capacité de raisonnement pour la reconnaissance de contextes et la prise de décision, l'adaptation et l'extensibilité afin d'assurer la pérennité, l'évolutivité du système par rapport à l'évolution de l'environnement ambiant et des usagers, et enfin la sécurité et la protection de la vie privé des usagers. La mise en œuvre de ces fonctionnalités nécessite de s'attaquer à plusieurs défis que nous analysons dans ce qui suit :

Auto-localisation

L'intelligence ambiante et la robotique ubiquitaire reposent sur l'identification et la localisation des personnes et des objets à l'intérieur d'espaces divers (maisons, bâtiments, à l'extérieur dans des espaces publics ou privés). Il est en effet primordial que les objets communicants (artefacts intelligents, robots, etc.) présents dans ces espaces soient capables d'identifier leurs positions, sans dépendre systématiquement d'un service central de calcul de position. Cette information de localisation est utile, par exemple, pour reconnaitre un contexte, analyser le comportement d'une personne ou pour choisir le service le plus approprié (proximité du service vis-à-vis de la position courante de l'utilisateur, préférences de l'utilisateur, etc.). Les systèmes de localisation disponibles actuellement dépendent le plus souvent d'un service central qui permet de calculer les coordonnées de localisation. Ils dépendent aussi fortement des contraintes matérielles des technologies utilisées. Il est alors nécessaire de combiner plusieurs technologies de localisation pour parvenir à une meilleure estimation de la position d'un objet. L'auto-localisation reste un challenge technologique car il n'existe pas à l'heure actuelle de solution fiable et bon marché.

Auto-exploration et cartographie sémantique

Il s'agit de la capacité du robot à explorer son environnement, identifier tout objet présent, comprendre sa description, ses fonctionnalités et exploiter ces connaissances pour effectuer une tâche donnée. Pour ce faire, le robot doit être capable de construire dynamiquement des cartes sémantiques en corrélant la description des objets avec celle de la cartographie physique. Il est aujourd'hui admis par la communauté robotique l'importance pour un robot autonome de disposer de connaissances sémantiques, et d'effectuer des raisonnements à un niveau sémantique (symbolique) pour mieux comprendre son environnement, reconnaitre un contexte, analyser une scène, comprendre le comportement humain, planifier une action donnée, etc. D'une manière générale, les ontologies spatiales et les cartes sémantiques peuvent être utilisées pour accroître les capacités et les performances de planification, de navigation et de manipulation, dans des scénarii complexes du monde réel. Pour un robot mobile, une carte sémantique est une carte qui comporte, en plus des informations spatiales (2D ou 3D) utilisées pour la navigation, des annotations sémantiques relatives aux caractéristiques des

entités de l'environnement du robot, c'est-à-dire, les objets, les fonctionnalités, les évènements, les relations, etc. [97].

Sensibilité et adaptation au contexte

Le paradigme de la sensibilité au contexte en informatique ubiquitaire a été introduit pour la première fois par Schillit qui considère que le contexte est basé sur trois aspects : La localisation de l'humain (Where are you ?), les identités des personnes avec qui il se trouve (Who are you with ?) et les objets qui l'entourent (what resources are nearby ?) [130]. Autour de ce paradigme, on trouve d'autres appellations, comme : La conscience du contexte (context awareness) ou l'adaptation au contexte. La définition de *Schillit* a été par la suite enrichie et adaptée par les chercheurs en fonction de leurs applications. Parmi les nombreuses définitions proposées dans la littérature, celle de *A.K. Dey* est la plus citée [35]. Elle stipule que le contexte couvre toute information pouvant être utilisée pour caractériser la situation d'une entité. Une entité peut être une personne, un lieu ou un objet qui est considéré comme pertinent pour l'interaction entre un utilisateur et une application, incluant l'utilisateur et les applications. Le contexte peut aussi concerner plusieurs modalités : (i) les profils des personnes et des objets présents dans un lieu ainsi que les évènements associés comme par exemple : La personne déplace un objet, la personne entre dans une pièce ; (ii) les moyens de communications et leurs qualités (fiabilité, bande passante, sécurité, etc.), les équipements et artefacts dont disposent les usagers (bracelet, montre, etc.) ; (iii) les moyens d'interaction avec l'usager (écrans, sons, vidéos, texte, etc.), les services simples ou complexes qui peuvent être fournis par des équipements ou des robots de services (télé-vigilance, assistance cognitive, identification de comportements sur le long terme, etc.).

D'une manière générale, on distingue deux types de connaissances contextuelles : (i) celles qui sont directement observables à partir d'organes de mesure (capteur de luminosité, système de localisation, sonde de mesure de la qualité d'un lien radio, etc.) et (ii) celles qui sont non-observables mais qui peuvent être déduites (par inférence, par classification, etc.). Le choix des capteurs et des sources d'informations contextuelles doit tenir compte des modes d'interaction avec les usagers et il est communément admis qu'il n'est pas nécessaire de tout savoir pour pouvoir aider et interagir de manière pertinente avec l'utilisateur [108]. Il s'agit de déterminer quels sont les capteurs appropriés dans une configuration donnée, de manière à ne pas suréquiper l'environnement. En plus de s'adapter au contexte lié aux activités quotidiennes des usagers, le système doit être capable d'estimer l'intention de l'utilisateur de manière transparente lorsque ce dernier est en train ou compte entamer une activité. Par conséquent, pour fournir des services intelligents sensibles au contexte, un système à intelligence ambiante ne doit pas seulement s'appuyer sur la localisation et l'identification des objets ou des personnes présents dans un lieu mais il doit aussi être conscient du contexte global, incluant l'environnement et les personnes.

Un système sensible au contexte est un système qui possède des capacités de réaction, d'adaptation et de reconfiguration face à des changements détectés dans l'environnement, dans l'intention de l'utilisateur et/ou de ses activités courantes. Pour *Dey*, l'un des objectifs principaux de l'acquisition du contexte est de déterminer par inférence ce que fait ou tente de faire l'utilisateur à un moment donné pour lui proposer des services et l'aider au mieux dans son activité. Les fonctions d'adaptation au contexte peuvent concerner : (i) la présentation de l'information : Des applications utilisent les informations contextuelles pour adapter la présentation de l'information ou modifier les possibilités d'interaction en proposant les actions les plus pertinentes à l'utilisateur comme par exemple : Présenter un choix d'imprimantes proches à un utilisateur ; montrer à l'utilisateur sa position géographique sur une carte ; (ii) l'exécution de services : Des applications exécutent automatiquement une ou plusieurs fonctions pour aider l'utilisateur. Par exemple, l'application peut délivrer un message ou une note lorsque l'utilisateur se trouve à un endroit particulier.

Récemment, et comme en témoigne les travaux menés dans la cadre du projet européen CA-Robocom (Robot Companions for Citizens) [7], le paradigme de la sensibilité au contexte a été repris par les roboticiens avec comme défi majeur le développement de robots sensibles, c'est à dire des robots exprimant des émotions et qui sont en même temps dotés d'une conscience de soi, des humains et de l'environnement.

Raisonnement

Un système à intelligence ambiante doit avoir des capacités cognitives avancées pour interpréter le contexte, reconnaître les activités et les intentions des usagers, et prendre des décisions adéquates en fournissant les services d'assistance les plus adaptés à la situation. Des techniques de raisonnement numérique et symbolique doivent être étudiées et adaptées pour doter le système de capacités cognitives avancées. Ainsi, l'inférence à base de règles de la logique du premier ordre peut être utile pour l'agrégation de connaissances contextuelles de bas niveau afin de transformer ces dernières en connaissances de haut niveau concernant le contexte. Un système d'apprentissage peut être utile pour reconnaitre des activités à partir de données multi-sources en tenant compte de contraintes telles que l'incertitude des données. Un système d'inférence basé sur les logiques de description peut aussi être exploité pour détecter des relations sémantiques entre les connaissances contextuelles. La prise de décision est en général suivie par l'exécution d'actions (services pro-actifs) comme par exemple :(i) ajuster la température et l'aération en utilisant les actionneurs disponibles dans l'habitation (ii) notifier à l'usager des commentaires, des suggestions ou des alertes instantanées ou en différé, à l'aide

7. http://www.robotcompanions.eu/project

d'interfaces multimodales personnalisées.

Intéropérabilité, adaptabilité et évolutivité

Bien que les systèmes ubiquitaires soient étudiés depuis une vingtaine d'années, la mise en œuvre de techniques d'interopérabilité reste toujours d'actualité pour traiter principalement les problèmes liés à l'hétérogénéité des environnements à intelligence ambiante. L'hétérogénéité pose le problème de partage des données et indirectement des connaissances contextuelles. Elle concerne d'une part, l'incompatibilité des moyens de communication (couches protocolaires), et d'autre part, les différents modèles représentant les données. L'émergence des intergiciels a permis de simplifier la mise en œuvre des systèmes répartis tout en masquant l'hétérogénéité des différents équipements et réseaux utilisés. En effet, les services d'intelligence ambiante interagissent avec les capteurs et les actionneurs à travers des intergiciels qui offrent un support de communication interopérable permettant de simplifier l'intégration et la perception abstraite des entités de l'environnement. Plusieurs intergiciels et protocoles d'interaction ont été standardisés pour le domaine de l'informatique ubiquitaire et l'intelligence ambiante, telles que UPnP, DLNA, DPWS, COAP, etc. Ces derniers reposent sur l'usage des protocoles de l'internet comme infrastructure de communication. Les intergiciels pour l'informatique ubiquitaire se déclinent en deux grandes catégories principales :

-La première permet de masquer l'hétérogénéité des capteurs, des actionneurs, des équipements et des couches protocolaires de communication, en offrant des interfaces standardisées d'accès aux sources d'informations contextuelles. Ces intergiciels offrent des fonctions de mesure, de pré-traitement et de représentation structurée de données numériques telles que la température, la pression atmosphérique, etc. À ces structures de données sont ajoutées des méta données, telles que des estampilles de temps, de lieux où les mesures ont été effectuées, etc.

-La deuxième catégorie fournit des modèles et des mécanismes élaborés de représentation, de gestion et de partage des connaissances entre des applications et services ubiquitaires hétérogènes. Du point de vue des applications ubiquitaires sensibles au contexte, l'exploitation des ontologies dans les intergiciels permet de garantir une certaine interopérabilité sémantique des connaissances du contexte. Il s'agit d'une part, de faciliter le partage d'une représentation standard et intelligible de la sémantique des connaissances contextuelles et d'autre part, d'automatiser les processus de découverte, d'extraction et de raisonnement sur ces connaissances contextuelles.

Enfin, garantir la pérennité des services d'intelligence ambiante et l'évolution de tout l'environnement d'intelligence ambiante pour accompagner les besoins des usagers, peut être vu comme autre conséquence positive de l'utilisation des intergiciels. Ces derniers doivent disposer de fonctions d'adaptation prenant en

compte la dynamicité de l'environnement en termes d'apparition ou disparition de services ou d'équipements, d'intégration de nouvelles applications ou équipements, la prise en compte de la mobilité des usagers et d'éventuels réaménagements de l'espace ubiquitaire. Enfin, pour faciliter la mise en œuvre de nouvelles applications, les intergiciels doivent fournir des outils qui permettent de simplifier la mise en œuvre des services, en utilisant par exemple des techniques d'assemblage de composants logiciels ou des techniques de composition de services.

Sécurité et respect de la vie privée

La sécurité des systèmes informatiques et la protection de la confidentialité et de la vie privée des usagers sont deux caractéristiques importantes que doivent posséder les services d'intelligence ambiante. Les environnements d'intelligence ambiante peuvent comporter de multiple sources d'informations dont il faut contrôler et protéger l'accès tout en respectant les aspects éthiques et juridiques. Mettre en place des mécanismes fiables de contrôle d'accès aux services, et des systèmes de chiffrement et déchiffrement des données de la vie privée, sont des problématiques difficiles et complexes qu'il faut impérativement aborder pour permettre à des personnes dépendantes d'évoluer et d'interagir de manière sécurisée avec les services de l'intelligence ambiante sans atteinte à leur vie privée.

Système non-intrusif et adaptatif

Les personnes dépendantes ont généralement besoin de services d'assistance et de soins et en même temps, elles souhaitent maintenir leur indépendance aussi longtemps que possible. Un système à intelligence ambiante doit par conséquent être capable de s'adapter aux habitudes et besoins des usagers pour améliorer ou maintenir leur qualité de vie. En règle générale, la qualité de vie est une notion assez complexe, ambigüe et difficilement mesurable. Dans un environnement d'intelligence ambiante, elle concerne en général la perception de la personne et son appréciation vis-à-vis de la façon dont les services d'intelligence ambiante influent (ou améliorent) sur son état de santé physique ou psychologique, réduisent sa dépendance vis-à-vis d'une assistance externe, maintiennent ou améliorent ses relations sociales, etc. [116]. Enfin, un aspect important qu'il convient de prendre en compte concerne le problème de l'acceptabilité de ces technologies d'assistance au quotidien.

2.5 Discussion et objectifs de la thèse

Dans cette thèse, nous nous focalisons sur la problématique de la représentation sémantique des connaissances et du raisonnement dans le cadre des systèmes à intelligence ambiante et des robots ubiquitaires. Il s'agit d'étudier des solutions adaptées permettant d'améliorer les fonctions cognitives de ces systèmes pour la gestion du contexte. Nous visons le développement de modèles sémantiques

qui soient suffisamment expressifs et génériques, pour permettre une description symbolique à la fois pertinente et riche de ces systèmes. Ces modèles doivent permettre d'une part, la reconnaissance de contextes et en particulier, les contextes non-observables et non-triviaux et d'autre part, de mettre en œuvre des fonctions d'adaptation. Un contexte non-observable peut être vu comme le résultat de l'agrégation d'un ensemble de connaissances contextuelles observables ou explicites. Nous considérons ici deux cas : L'exploitation d'observations courantes et l'exploitation de l'historique des observations. Les modèles proposés doivent permettre par exemple de décrire les activités de l'utilisateur allant des plus élémentaires (se lever/s'assoir, marcher, allumer le four, etc.) aux plus complexes telles que préparer un repas. Ces dernières peuvent être vues comme des séquences d'évènements associés à des activités élémentaires. Les objectifs de cette thèse concernent la proposition de deux modèles sémantiques répondant aux contraintes imposées par les systèmes à intelligence ambiante et les robots ubiquitaires, et leurs validations à travers la mise en œuvre de scnéarii réalistes en utilisant la plateforme ubiquitaire développée au laboratoire LISSI.

Le premier modèle sémantique proposé vise la représentation explicite des entités physiques ou logiques (humain, robot, capteurs, actionneurs, services web, etc.), et des connaissances contextuelles associées à ces entités ; l'objectif étant d'exploiter ces connaissances dans les services sensibles au contexte. Pour satisfaire aux contraintes d'expressivité et d'extensibilité, il s'agit d'aboutir à partir de ce modèle, à un modèle d'ontologie pour la description d'un environnement à intelligence ambiante. Par ailleurs, il est nécessaire d'associer à ce modèle un langage de règles pour modéliser le processus de raisonnement réactif sur les connaissances contextuelles avec deux objectifs : La reconnaissance implicite de contextes et la fourniture de services sensibles au contexte. Les contraintes imposées par les applications d'intelligence ambiante en termes de dynamicité du contexte et de cohérence dans la prise de décision, exigent des modes de raisonnement garantissant les propriétés essentielles suivantes : Décidabilité, non-monotonie du raisonnement. Par cette approche, l'objectif est de pallier aux problèmes rencontrés dans les ontologies du web sémantique qui s'appuient sur le raisonnement monotone et qui sont par conséquent inadaptées pour prendre en compte dynamiquement des changements de contexte ou pour inférer des contextes non-observables.

Le deuxième modèle vise, quant à lui, à compléter le modèle précédent en termes d'expressivité pour la représentation de contextes non-triviaux liés au temps. Il s'agit ici de pallier aux inconvénients des approches ontologiques du Web sémantique en utilisant des relations n-aires pour la représentation des connaissances contextuelles. Sémantiquement, ce modèle doit permettre une représentation narrative des événements. Associé à des mécanismes d'inférence appropriés, ce modèle vise également à établir des relations sémantiques (causalité, but, etc.) implicites entre les évènements qui se produisent dans l'environnement. Plus précisément, il s'agit de modéliser les interdépendances entre contextes, et

d'exprimer des connaissances de type : qui est l'initiateur de l'événement/action ? dans quel contexte un événement est observé ou une action est exécutée ? quelle est l'entité qui a subit l'action ? etc. L'objectif est d'enrichir l'interprétation du contexte en vue d'assurer une meilleure adaptation à ce dernier.

Le troisième objectif de cette thèse concerne la mise en œuvre et la validation des modèles sémantiques proposés. L'évaluation de ces modèles ne doit pas se limiter à la mise en œuvre d'un environnement de simulation élémentaire avec des outils de tests unitaires, mais elle doit être réalisée dans un espace d'intelligence ambiante réel. Ce dernier doit être composé d'un ensemble de capteurs, actionneurs, robot, etc. L'environnement d'expérimentation doit reposer sur un intergiciel permettant de faciliter l'intégration de technologies matérielles et logicielles hétérogènes, et la mise en œuvre de services d'intelligence ambiante sensibles au contexte.

Chapitre 3

Représentation des connaissances et raisonnement : État de l'art

Sommaire

3.1 Introduction

Dans ce chapitre, nous présentons un état de l'art sur les formalismes de représentation symbolique des connaissances et les modèles de raisonnement associés. Nous faisons tout d'abord un rappel sur les concepts de la logique du premier ordre et du raisonnement logique qui représentent les briques de base communes à tous les systèmes de représentation et de raisonnement symbolique. Nous étudions ensuite les différents modèles de représentation de connaissances et de raisonnement à base d'ontologies, puis nous passons en revue les principaux travaux exploitant la représentation par ontologies, dans le domaine de l'intelligence ambiante et de la robotique. La dernière partie du chapitre est consacrée à l'étude des principales approches de modélisation, de représentation temporelle et de manipulation des connaissances dynamiques.

3.2 Les prédicats et les propositions

En logique du premier ordre, la représentation des connaissances et le raisonnement, requièrent l'utilisation de prédicats et de propositions pour représenter les axiomes, les concepts et les relations entre concepts, ainsi que des règles d'inférence. Une proposition est une affirmation munie d'un sens. Elle correspond à une assertion d'un énoncé élémentaire qui a l'une des deux valeurs de vérité : vraie ou fausse. Par exemple, l'assertion "John est dans sa chambre" est vraie tandis que l'assertion "John est dans la cuisine" est fausse. La logique propositionnelle est le formalisme logique le plus simple et le plus décidable [1] dans lequel un énoncé E est dit satisfaisable (ou consistant) si sa valeur de vérité est vraie dans une interprétation I, formellement $I \vDash E$ [93]. Ainsi, l'énoncé $(p \vee q)$ est satisfaisable pour une interprétation si les valeurs de p et de q dans la table de vérité sont comme suit :{ v (p) = vrai, v (q) = vrai}, {v (p) = faux, v (q) = vrai} et { v(p) = vrai, v (q) = faux}. À l'opposé, si un énoncé E n'admet pas une interprétation qui le rend vrai, alors il est dit non satisfaisable. La conjonction d'une proposition et de son complément : $(p \wedge \neg p)$ est un exemple d'énoncé n'admettant pas d'interprétation.

Les prédicats peuvent être considérés comme une généralisation des propositions et forment une logique plus expressive que la logique des propositions. En effet, l'utilisation de symboles correspondant aux quantificateurs universel et existentiel, fonctions et variables, permet d'augmenter l'expressivité de la logique du premier ordre. Les fonctions peuvent être d'arité quelconque, et les variables sur lesquels s'appliquent les quantificateurs, représentent les objets de l'univers. Pour exprimer l'énoncé suivant : "Tout espace de la maison est équipé d'un capteur", nous pouvons par exemple utiliser les prédicats : Space_Region(X), equippedWith(Y,X) and Sensor(Y).

1. Un problème est décidable si une machine de Turing peut le résoudre en un nombre fini d'étapes.

En logique des prédicats, l'énoncé précédent s'écrit :
f1 : $\exists$ y (Sensor(Y) $\wedge$ ($\forall$ X.equippedWith (X,Y) $\wedge$ Space_Region(X))) ;

En langage naturel, l'énoncé f1 peut être exprimé comme suit : "Il n'existe pas d'espace qui n'est pas équipé de capteurs". En logique des prédicats, cet énoncé s'écrit : $\nexists$X. equippedWith (X,Y) $\wedge$ Space_Region(Y) ;

Une des caractéristiques fondamentales de la logique du premier ordre est qu'elle est monotone. Ce qui signifie qu'une assertion est toujours considérée comme vraie et quelle ne peut jamais être réfutée. Formellement, la monotonie est définie comme suit : Soit {*P1, ..., Pn, R, Q*} un ensemble de formules bien formées. $\forall$ *P1, ..., Pn, R, Q,* si à partir des formules *P1, ..., Pn*, la formule *Q* est déductible de l'ensemble (*P1, ..., Pn,R*), il est alors toujours possible de déduire la formule *Q*. En d'autres termes, si une formule est déductible d'un ensemble de formules, alors cette formule reste déductible si l'on ajoute à l'ensemble initial de nouvelles formules.

3.3 Représentation et raisonnement à base d'ontologies

3.3.1 Historique et définitions

Le terme "ontologie" désigne l'étude des propriétés générales de ce qui existe. Empruntée à la philosophie, ce terme est employé dans le domaine informatique pour la première fois par *John McCarthy* dans les années 80 [83], [13]. Ce dernier met en évidence le recouvrement qui existe entre les travaux sur les ontologies en philosophie et ceux liés à la définition des théories logiques en intelligence artificielle. De nombreuses définitions ont été données au terme "ontologie" mais elles sont assez similaires dans le fond. Les deux définitions les plus largement citées dans la littérature sont celles données par *Gruber* [47] et *Studer* [139]. *Gruber* définit une ontologie comme une spécification explicite d'une conceptualisation, qui correspond à une vue abstraite et simplifiée du monde que l'on veut représenter. Selon *Studer*, une ontologie est une spécification formelle compréhensible par la machine et explicite d'une conceptualisation partagée par un groupe d'individus.

Selon *Uschold et al.* [146], une ontologie constitue un support pour une description sémantique, structurée et formelle des concepts d'un domaine, et de leurs inter-relations. Elle permet de faciliter les échanges de connaissances d'une part, entre les usagers et les systèmes et d'autre part, entre les systèmes eux-mêmes. Les ontologies constituent donc un outil puissant pour la conceptualisation formelle d'un environnement et le partage de connaissances entre des systèmes hétérogènes. Les ontologies, sont souvent associées à des moteurs d'inférence pour raisonner sur les connaissances qu'elles représentent. Les ontologies peuvent être classées en deux types :

- Les ontologies légères : Ces ontologies ont une structure relativement simple, et sont généralement composées de concepts, et de taxonomies de concepts. Une

taxonomie de concepts est une hiérarchisation de concepts liés par des relations de subsomption [2]. Ces relations expriment des relations entre concepts de type Is-A (est-un) et entre les propriétés décrivant ces concepts ;
– Les ontologies riches : À la base, il s'agit d'ontologies légères complétées par des restrictions sémantiques concernant le domaine de description, comme par exemple, des contraintes de cardinalité [45] ;

À l'heure actuelle, les ontologies sont au cœur de nombreuses applications telles que la description sémantique de ressources multimédia et d'objets physiques (au sens du web des objets). Elles visent à faciliter le processus de recherche et de fouille de données, la mise en œuvre des services web sémantiques, et la conception de systèmes ubiquitaires. Outre le fait d'établir un consensus sur la description des concepts utilisés par les applications pour le partage de connaissances, les ontologies sont également associées à des mécanismes d'inférence permettant d'automatiser le raisonnement sur ces connaissances. Cependant, la représentation de connaissances et le raisonnement à base d'ontologies n'ont été rendu possible qu'avec d'une part, l'avènement du web sémantique et des langages et outils associés, basés sur XML et RDF et d'autre part, le développement de moteurs d'inférence exploitant ces formats de représentation. Le web sémantique est sans doute l'un des plus grands projets concernant l'utilisation des ontologies en tant que support pour l'interopérabilité sémantique entre des systèmes informatiques hétérogènes dans le web. À l'origine, l'objectif premier du Web sémantique était de mettre en œuvre des formalismes et des outils de représentation et de traitement des connaissances symboliques pouvant être utilisées par des agents logiciels pour rechercher, traiter et agréger automatiquement des connaissances hétérogènes multimodales disséminées dans le Web.

3.3.2 Fondements des langages d'ontologies

La plupart des langages de définition d'ontologies proposés dans l'état de l'art sont fondés ou s'inspirent des logiques du premier ordre et représentent les connaissances sous forme d'assertions (sujet, prédicat, objet). Ces langages sont typiquement conçus pour s'abstraire des structures de données et se concentrer essentiellement sur leur sémantique. Parmi les formalismes de définition d'ontologies, il existe d'une part, des langages de représentation structurés des ontologies tels que KL-ONE [19], LOOM [76], KIF-Ontolingua [17], Cycl [3], DAML [4], DAML-OIL [78], F-Logic, OWL, etc. et d'autre part, des formalismes permettant de décrire formellement les connaissances telles que, les graphes conceptuels ou les logiques de description. Dans ce paragraphe, nous présentons les langages de schémas (Frame) introduits en intel-

2. Ce terme, emprunté à la philosophie, tend à remplacer le terme "généralisation/spécialisation" dans le domaine des ontologies. Il constitue le principal élément pour la structuration d'une ontologie. On parle ainsi de "liens de subsomption". Par exemple, le concept *Etre_Humain* est une spécialisation du concept *Homme*. Formellement, Homme $\sqsubset$ *Etre_Humain*, signifie que le concept *Etre_Humain* subsume le concept *Homme*.
3. http://www.cyc.com/knowledgeserver
4. http://www.daml.org

ligence artificielle comme alternative à la représentation des connaissances basée sur les réseaux sémantiques. Ces derniers ont inspiré la plupart des travaux sur les formalismes et langages de définition des ontologies. Les schémas (Frames ou cadres) introduits par *Marvin Minsky*, correspondent à des structures de données hiérarchiques pour représenter une situation stéréotypée [91]. En effet, les schémas ont été proposées pour rassembler les connaissances pertinentes d'une situation donnée dans un seul objet appelé Frame au lieu de répartir ces connaissances à travers plusieurs axiomes. Les schémas possèdent un ensemble d'attributs (propriétés) appelés *slots*, de *facettes* et de *relations*. Les *slots* définissent une structure de données. Les facettes décrivent sémantiquement les valeurs possibles pour chaque *slot*. Enfin, les *relations* sont de type spécialisation/généralisation. Un Frame peut être vu comme un réseau de nœuds et de relations entre nœuds (réseau sémantique). Les connaissances supportées par les frames sont soit décrites explicitement ou bien déduites par inférence. Ces déductions s'effectuent principalement en exploitant l'héritage d'attributs à partir des frames-pères.

Le langage Cycl a été le premier langage d'ontologie basé dans sa version initiale sur les Frames puis sur la logique du premier ordre dans sa dernière version. Il a été utilisé pour décrire l'ontologie universelle Cyc. Cette dernière est composée de plus de 3 millions d'assertions (faits, axiomes et règles) concernant tous les aspects du monde. Les termes sont groupés en 43 thèmes (Temps, Relations spatiales, etc.).

Les réseaux sémantiques, introduits à l'origine par *Quillian* [111], utilisent des structures graphiques pour représenter les connaissances. Les réseaux sémantiques sont représentés par des graphes où les nœuds représentent des objets (concepts, situations, événements, etc.) et les arcs expriment des relations (liens) entre objets [69]. Ces relations sont de deux types :

1 : La relation *"est-sorte de"* entre concepts, par exemple, le concept *Siège* est-sorte de *Meuble* ;
2 : La relation *"est-un"* entre un concept et un individu, par exemple, une *chaise* est-un *Siège* ;

Le raisonnement dans les réseaux sémantiques consiste en des opérations de manipulation de graphes. Ainsi, l'utilisation de la notion de sous-classe/super-classe permet d'organiser les concepts comme une spécialisation/généralisation de concepts. Dans ce formalisme, les propriétés sont limitées à des propriétés primitives, alors qu'en général, les propriétés dans les schémas peuvent être plus complexes. La représentation graphique des réseaux sémantiques rend ces derniers facilement interprétables par l'humain, et exploitables par une machine.

3.3.3 Les graphes conceptuels

La représentation des connaissances à l'aide des graphes conceptuels s'appuie sur la description de concepts et de leurs relations [134]. Les connaissances du domaine d'application sont définies par les graphes canoniques associés à certains

types d'informations. Ces dernières sont associées à des rôles comme : 'AGT [5]', 'ORIG [6]', 'OBJ [7]', 'LOC [8]', etc. Dans un graphe conceptuel, un concept est formé de deux éléments : son type et son référent. Par exemple, une *Table* est un objet que l'on peut définir comme suit : *Objet :Table*, où *Objet* représente le type et *Table* le référent.

Une base de connaissances représentée en graphe conceptuel comprend plusieurs niveaux terminologiques : Le premier, appelé "support", est composé d'un ensemble de concepts et de relations. Ces deux ensembles forment une hiérarchie définissant une relation d'ordre [9] notée respectivement $\leq_C$ *et* $\leq_R$. Le second niveau terminologique, appelé "niveau assertionnel", est composé de faits, c'est-à-dire, les instances. Dans les graphes conceptuels, les relations servent à étiqueter les sommets des graphes ainsi que leur ordonnancement selon une relation de spécialisation/généralisation. Le support peut être enrichi par des règles et des contraintes pour introduire plus d'expressivité.

Mugnier et al. distinguent deux types de contraintes : contrainte positive et contrainte négative [94], [8]. Une contrainte positive est composée d'une condition et d'une obligation, comme par exemple : un espace est contrôlé par au moins une caméra. Une contrainte négative est composée, quant à elle, d'une condition et d'une interdiction comme par exemple : Une personne ne peut pas être à la fois assise et debout.

La modélisation à l'aide de graphes conceptuels s'effectue selon une représentation intentionnelle [10]. Rappelons que la définition d'une classe constitue l'intention de la classe et que les individus/instances de cette classe en constituent l'extension. La modélisation des connaissances s'effectue en deux étapes : (1) Définition du support permettant de spécifier les concepts et leurs relations. (2) Création des graphes pour modéliser les faits (instances) de l'application considérée. Nous présentons dans ce qui suit le formalisme des graphes conceptuels tel que défini par *Chein et al.* [26] et [28].

5. Agent intervenant de façon active, initiateur de l'action
6. Origine, source spatiale ou abstraite d'un événement
7. Objet, entité affectée ou subissant l'action
8. Localisation, indique le lieu de déroulement d'un événement
9. La relation d'ordre est définie selon le principe "est une sorte de" noté $\leq_C$. Considérons t_1 et $t_2 \in T_C$, tel que T_C est un ensemble ordonné de concepts. Si $t_1 \leqslant t_2$, alors t_1 est plus spécifique que t_2 et t_2 est plus général que t_1
10. Elle consiste à définir des classes et des relations, puis à instancier et à lier des individus/instances, contrairement à la représentation extensionnelle qui consiste à s'intéresser d'abord aux individus/instances, puis à regrouper ces individus/instances en fonction de caractéristiques communes et ainsi définir des classes d'individus/instances

Le niveau support ou le niveau terminologique

Un support est représenté par le quintuplet S= (T_C , T_R, σ, ψ, δ) où :

- T_C représente un ensemble ordonné de concepts. Son plus grand élément, noté $\top$, représente la catégorie la plus générale qui contient toutes les instances. Son plus petit élément, noté $\bot$, représente, quant à lui, la catégorie vide.
- T_R représente l'ensemble des relations partitionnées en sous-ensembles de relations de même arité. Ces relations, également ordonnées, sont de type "sorte-de".
- σ associe à chaque type de relation t d'arité n sa signature $\in T_C^n$;
- ψ représente l'ensemble des marqueurs individuels. Il s'agit d'un ensemble dénombrable qui comporte également un marqueur générique noté * permettant de représenter un individu non identifié ;
- δ est une application de ψ dans $T_C \setminus \{\bot\}$ dite de conformité, qui associe à chaque individu un concept.

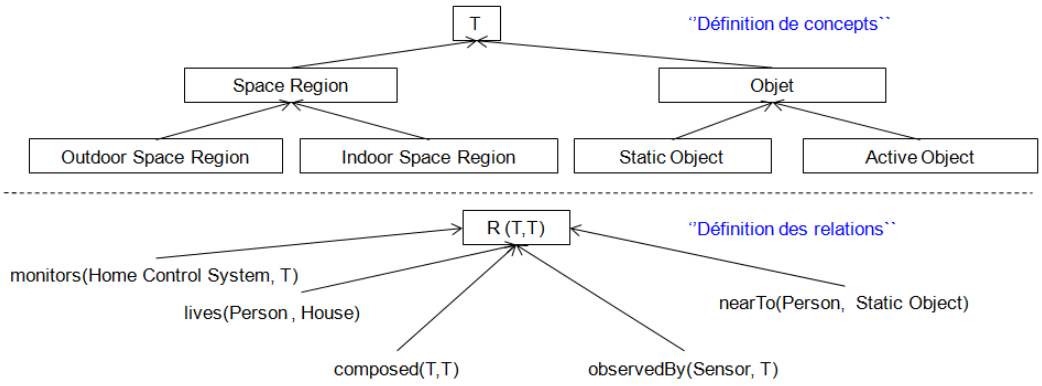

FIGURE 3.1 – Exemple de support d'un graphe conceptuel.

La figure 3.1 illustre un exemple de support de graphe conceptuel incluant un ensemble de relations binaires. Le concept *Indoor_ Space_ Region* est une sorte de concept *Space_ Region* ; on dit que, le concept *Indoor_ Space_ Region* est subsumé par le concept *Space_ Region*. La relation *nearTo*(Person, Static_Object) a une signature à deux arguments : Le concept *Person* et le concept *Static_ Object*.

Le niveau assertionnel

Ce niveau permet de décrire des individus par des graphes conceptuels en utilisant la terminologie du support. Ainsi, un individu peut être représenté par une combinaison de relations définies dans le support. La figure 3.2 représente les connaissances factuelles suivantes : (i) L'objet *Kitchen* est contrôlé par une *Camera* (fait

(a)) ; (ii) La personne nommée *David* est localisée devant un objet statique *Stove* (fait (b)). Ces faits sont instanciés à partir du support illustré, figure 3.1,

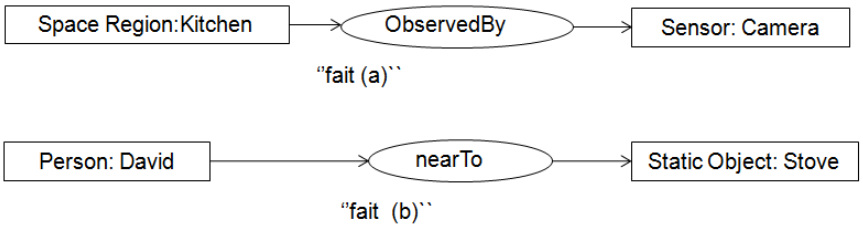

FIGURE 3.2 – Exemples de faits exprimés en graphe conceptuel.

Exemple. Dans cet exemple, il s'agit de modéliser en graphe conceptuel le déplacement d'un *robot* d'un lieu initial (*Living_Room*) vers un lieu final (*Kitchen*). Dans la figure 3.3, les individus *Living_Room, Kitchen* sont associés au concept *Space_Region*, tandis que l'individu *Home_Control_System* est associé au concept *Control_System*. La relation *agent* indique l'agent qui exécute l'action *move*. La relation *deparature* indique la position courante du *robot*, la relation *dest* la nouvelle destination du robot et la relation *object* l'objet à déplacer.

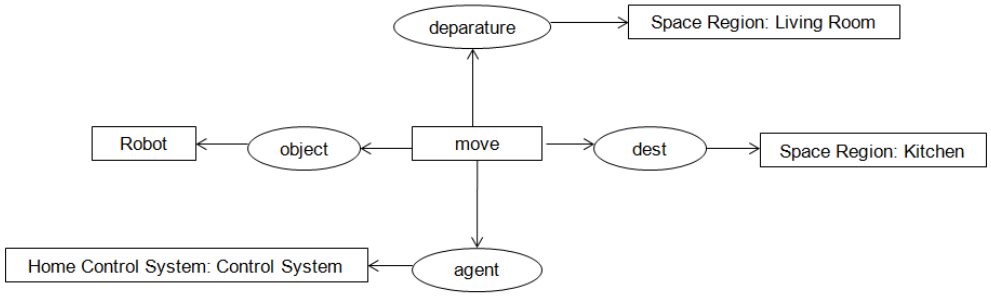

FIGURE 3.3 – Le graphe canonique de l'action Déplacer.

Pour des raisons pratiques, les graphes conceptuels sont aussi exprimés à l'aide de notations linéaires appelées *CGIF* (*Conceptual Graph Interchange Format*). Dans cette représentation, les concepts sont représentés entre crochets et les relations entre parenthèses. Le graphe conceptuel illustré, figure 3.3 peut être exprimé en forme linéaire comme suit :

[move :*x][Home_Control_System : Control_System][Object :Robot][Space_Region :Living_Room][Space_Region :Kitchen] (agent ?x Home_Control_System) (object ?x Robot)(deparature ?x Living_Room) (dest ?x Kitchen).

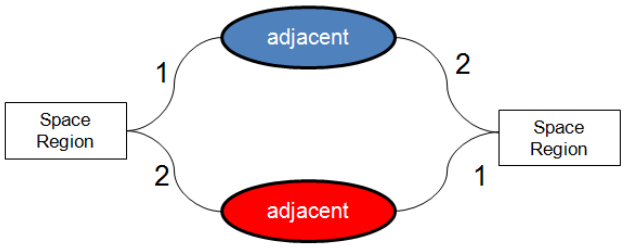

FIGURE 3.4 – Exemple de règle exprimée en graphe conceptuel [9].

3.3.4 Raisonnement à partir de graphes conceptuels

Nous reprenons ici la définition d'une règle selon *Salvat et Mugnier* [122] [123]. Ces derniers définissent une règle R comme un graphe conceptuel bicolore, définit sur un support S. Ainsi, deux sous-graphes sont engendrés à partir des deux couleurs attribuées aux sommets de R. Le premier sous-graphe P_R, engendré par les sommets de la première couleur, représente la *précondition* de R (*précond* (C_R)). Le sous-graphe, engendré par les sommets de la deuxième couleur, représente la conclusion de R (*cond(R)*). Une règle R est applicable à un graphe conceptuel si et seulement il existe une projection de *précond(R)* dans le graphe G. La règle illustrée figure 3.4, signifie que si un *espace* x est adjacent à un *espace* y alors y est adjacent à x [9]. Dans cette règle, les sommets de l'hypothèse sont colorés en bleu, et ceux de la conclusion en rouge.

Le raisonnement sur les graphes conceptuels s'effectue à l'aide d'opérations de graphes, basées sur un morphisme de graphe appelé projection [27]. La projection d'un graphe conceptuel H=(C_H, R_H, U_H, $\in_H$) dans un graphe conceptuel G=(C_G, R_G, U_G, $\in_G$), définie sur un même support S, est un morphisme de graphe biparti, c'est-à-dire, un couple d'applications $\Pi = (\rho_C : C_H \longrightarrow C_G , \rho_R : R_H \longrightarrow R_G)$ respectant les contraintes imposées par le support, comme la restriction à la fois sur les sommets relations et les sommets concepts. Cette opération de projection peut être décomposée en un ensemble d'opérations élémentaires de spécialisation/généralisation entre graphes [135]. La plupart des raisonnements réalisés à partir des graphes conceptuels sont reconnus comme des problèmes NP–complets [26] [10].

3.3.5 Logiques de description

Les réseaux d'héritage structurels (Structured Inheritance Networks) (SIN) [11]" ont été introduits dans les années 70 par *Brachman* [21] [22] pour lever les ambiguïtés des réseaux sémantiques et des schémas. Les trois idées suivantes, mises en avant dans le travail, de *Brachman*, ont largement façonné le développement

11. (http ://www.ontotext.com/factforge/sin)

ultérieur des logiques de description :

- Les blocs de construction syntaxique de base sont : Les concepts atomiques (prédicats unaires), les rôles atomiques (prédicats binaires) et les individus (constantes) ;
- La puissance d'expressivité des langages des logiques de description est restreinte du fait qu'ils n'utilisent qu'un ensemble restreint de constructeurs épistémologiquement pour la construction de concepts complexes et de rôles ;
- Des connaissances implicites sur les concepts et les individus peuvent être inférées automatiquement à l'aide de procédures d'inférence, en particulier, les liens de subsomption entre les concepts et entre les relations (propriétés) ;

KL-ONE [23], LOOM [81] et KRIS [7] sont les premiers projets phares sur les logiques de descriptions. KL-ONE est considéré comme l'ancêtre des systèmes utilisant les logiques de description [152]. Il a permis la transition des réseaux sémantiques vers une description (terminologie) logique bien-fondée. KL-ONE a ainsi permis d'introduire la plupart des notions clés des logiques de description telles que les notions de concept, de rôle et la façon dont les concepts et les rôles doivent être liés.

Ces dernières années, une grande communauté de recherche s'est constituée autour du développement d'outils de représentation et de raisonnement avec les logiques de description, notamment pour la mise en œuvre des fondements du web sémantique. En effet, les logiques de description permettent de représenter de manière structurée et formelle les connaissances d'un domaine par des concepts, des rôles et des individus. En général, la représentation des connaissances en logique de description s'appuie sur trois éléments principaux :

- La représentation des connaissances intentionnelles est définie sous la forme d'une terminologie contenue au niveau du composant TBox ;
- Les assertions représentent les individus (instances) du domaine ; ces individus sont définis au niveau du composant ABox communément nommé *niveau factuel* ;
- Un ensemble de constructeurs (conjonction, disjonction, négation, restriction d'une valeur, quantificateur existentiel, etc.) existe pour combiner des concepts et des rôles ;

Les logiques de description se déclinent en plusieurs langages se distinguant par leurs degrés d'expressivité et de complexité. Cette classification est basée principalement sur le choix du constructeur. Le tableau 3.2 présente les différents constructeurs utilisés dans les langages de description.

La logique $\mathcal{AL}$ ("Attributive language") est dite logique minimale par le fait qu'elle soit la moins expressive des logiques, tableau 3.1. Comme le montre le tableau 3.1, la négation d'un concept, telle que $\neg Women$, ne peut être utilisée que pour un concept atomique. De même le quantificateur existentiel ne peut être

appliqué qu'avec le top concept ($\top$). Par exemple, avec le rôle atomique *wears*, il est possible d'exprimer comme suit le fait qu'une personne porte un capteur : $Person \cap \exists$ wears.$\top$.

C,D $\rightarrow$ A (concept atomique)
$\mid \bot$ *bottom concept*, représente le concept vide.
$\mid \top$ *top concept*, représente les individus (instances) des concepts associés aux entités de l'environnement.
$\mid \neg A$ négation
$\mid C \cap D$ intersection de concepts
$\mid \exists R.\top$ restriction sur une valeur
$\forall R.C$ quantificateur existentiel

TABLE 3.1 – La logique de description $\mathcal{AL}$

Langages		Constructeurs	Concepts
$\mathcal{AL}$	$\mathcal{FL}0$	Noms de concepts atomiques	A, C
		Rôles atomiques	R_1, R_2
		Intersection (conjonction de concepts)	A $\cap$ C
		Quantificateur universel (restriction de valeur)	$\forall R.C$
	$\mathcal{FL}^-$	Quantificateur existentiel	$\exists R$
		Top (subsume tous les concepts)	$\top$
		Bottom (n'a pas d'instance)	$\bot$
	$\mathcal{C}$	Négation	$\neg A$
	$\mathcal{U}$	Union (Disjonction)	$A \cup C$
	$\mathcal{E}$	Quantificateur existentiel typé	$\exists R.C$
	$\mathcal{R}$	Conjonction de rôles	R_1, R_2
	$\mathcal{N}$	Cardinalités	$(\geq nR)(\leq nR)$
	$\mathcal{H}$	Hiérarchie de rôles	$R_1 \subset R_2$
	$\mathcal{I}$	Rôle inverse	R^-
	$\mathcal{Q}$	Restriction de nombres (Qualified numbre restriction)	$(\geq nR.A)(\leq nR.A)$

TABLE 3.2 – Correspondances entre les constructeurs des différents langages de description

Il existe trois moyens d'augmenter l'expressivité de la logique $\mathcal{AL}$:
- Ajouter des constructeurs de rôles ;
- Ajouter des constructeurs de concepts. La logique $\mathcal{ALUE}$ n'est rien d'autre que la logique $\mathcal{AL}$ enrichie des symboles union $\cup$ et quantificateur existentiel $\mathcal{E}$;
- Définir des contraintes sur l'interprétation des rôles ;

La logique $\mathcal{AL}$ peut être étendue à l'aide des constructeurs suivants : $[\mathcal{U}][\mathcal{E}][\mathcal{N}][\mathcal{C}][\mathcal{I}][\mathcal{F}][\mathcal{H}][\mathcal{Q}][\mathcal{O}]$. Pour exprimer les concepts par énumération d'individus nommés, il suffit d'ajouter la lettre $\mathcal{O}$ à la logique $\mathcal{AL}$. La négation complète d'un concept est exprimée en ajoutant la lettre $\mathcal{C}$ à la logique $\mathcal{AL}$. Le rôle inverse d'un rôle et l'inclusion entre rôles sont représentés respectivement par les lettres

$\mathcal{I}$ et $\mathcal{H}$. Les constructeurs $\mathcal{N}, \mathcal{F}$ et $\mathcal{Q}$ complètent la logique $\mathcal{AL}$ pour exprimer la contrainte de cardinalité sur un rôle.

Pour illustrer la modélisation d'un environnement de type maison intelligente en logique de description, considérons un habitat composé des espaces suivants :*Bath_ Room, Room, Kitchen, Hall*. Ces espaces sont vus comme des individus appartenant au composant ABox. Supposons que le composant TBox contienne les concept *Sensor* et *Space_ Region*, qui est une abstraction (concept parent) des espaces énumérés ci-dessus, et considérons les rôles suivants : *equipped-With, monitored*.

Les expressions (1),(2),(3) ci-dessous, expriment en logique de description les axiomes suivants : Tout espace est équipé d'un capteur (1), ce qui revient à dire par l'axiome (2) qu'il n'existe pas d'espace non équipé d'un capteur. Selon l'axiome (3), un espace est dans la zone de couverture d'un capteur de type caméra.

(1) $\forall Space\_Region.(Space\_Region \cap \exists$ equippedWith.Sensor) ;

(2) $Space\_Region \cap \exists$ (equippedWith.Sensor).$\perp$;

(3) monitored $\equiv Space\_Region(\exists$ equippedWith $(Sensor \cap$ Camera)) ;

3.3.6 Raisonnement en logique de description

Les inférences en logique de description consistent à manipuler les éléments des composants TBox et ABox en effectuant les opérations suivantes :

- Subsomption : Cette opération permet d'établir une hiérarchie de concepts basée sur le principe de généralisation/spécialisation. Considérons par exemple, les deux axiomes suivants :

  (4) monitoredSpaceRegion $\equiv Space\_Region \cap \exists$.equippedWith.Sensor ;

  (5) monitoredSpaceRegion $\sqsubseteq Space\_Region$;

  L'axiome (4) signifie ici qu'un espace, équipé d'au moins un capteur, est un espace contrôlé *(monitoredSpaceRegion*). Ici, la subsomption permet d'inférer que tout concept de type *monitoredSpaceRegion* est un concept *Space_ Region*. Par conséquent, l'axiome : *Bath_ Room* $\sqsubseteq$ *monitoredSpaceRegion* signifie implicitement que le concept *Bath_ Room* est un *Space_ Region*, même s'il n'a pas été explicitement exprimé dans le composant TBox ;

- Satisfaisabilité : Un concept A est dit satisfaisable par rapport à une terminologie définie dans le composant TBox, s'il existe toujours une interprétation de A. Autrement dit, I(A)$\neq \emptyset$. Par exemple, l'axiome : *monitoredSpaceRegion* $\sqcap\neg$ *monitoredSpaceRegion* exprime le fait qu'un espace ne peut être à la fois contrôlé et non-contrôlé. Autrement dit, l'intersection d'un concept et de son complément est toujours un ensemble vide=$\emptyset$;

– Equivalence : Deux concepts A et B sont équivalents si pour tout axiome du composant TBox, il existe une interprétation I telle que I(A)≡ I(B) ;

– Disjonction : Deux concepts A et B sont disjoints si les deux concepts sont sémantiquement différents par rapport au composant TBox, et on écrit $I(A) \sqcap I(B) = \emptyset$;

Les inférences sur le composant du niveau factuel ABox impliquent les opérations suivantes :

1. Le contrôle de cohérence : Il s'agit de vérifier que les assertions définies dans le composant ABox sont cohérentes par rapport au composant TBox ;
2. La vérification d'instance : Elle se traduit par la vérification que chaque instance respecte la définition de son concept.
3. La vérification de rôle : Elle consiste à vérifier si les rôles d'une instance dans une assertion R(a,b) telle que a et b correspondent à des individus ;
4. La réalisation du composant ABox : Elle consiste à trouver le concept le plus spécifique pour chaque instance ;

3.4 Langages du web sémantique

3.4.1 RDF

Le langage RDF est basé sur la conjonction de trois primitives : classe, propriété et instance, pour décrire toute information sous la forme d'un triplet. Un triplet RDF est une association de trois composants : Le Sujet qui correspond à une ressource Web, le Prédicat binaire qui correspond à une propriété et l'Objet qui représente une valeur de cette propriété. Un prédicat RDF a pour rôle de décrire la relation entre un Sujet et un Objet ; ces derniers représentant respectivement le domaine et le type (en Anglais : range) de l'objet. Chaque ressource web est représentée par un identifiant unique appelé URI (Universal Ressource Identifier) [16]. À titre d'exemple, l'énoncé : "Un système éteint la cuisinière" peut être représenté par le triplet composé des deux ressources : *Home_ Control_ System et Stove*, et du prédicat *switchOff*, figure 3.5.

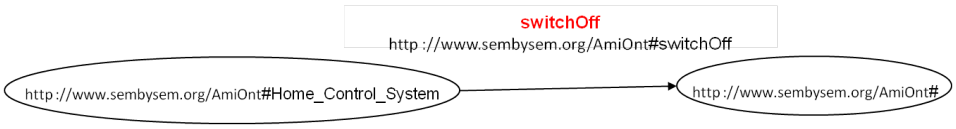

FIGURE 3.5 – Représentation graphique en RDF de l'énoncé : *Le système éteint la Cuisinière.*

3.4.2 RDF-S

Bien que RDF soit un outil conçu pour la représentation et le partage des informations, il n'offre cependant aucun moyen permettant de donner une sémantique aux informations partagées. Pour y remédier, le langage RDF est généralement associé à un vocabulaire prédéfini nommé "RDF Vocabulary Description Language". On parle dans ce cas du langage RDF Schema (ou RDF-S) [24]. Le langage RDF-S permet de décrire sémantiquement les informations échangées. Dans cette extension, la relation d'héritage entre classes ou entre propriétés n'est autre qu'une taxonomie des termes organisés de manière hiérarchique. Le langage RDF-S permet également de définir des restrictions sur les valeurs de propriétés offrant ainsi un moyen d'effectuer de simples inférences comme l'appartenance à une classe ou à une sous-classe. La vérification des valeurs de propriétés et des relations de sous-propriétés est réalisée à l'aide du principe de généralisation/spécialisation entre propriétés [52]. Par exemple, il est possible d'exprimer à l'aide du langage RDF-S que la ressource *Home_ Control_ System* est une classe, ce que ne permet pas le langage RDF. Les lignes 4 et 5 du tableau 3.3 permettent de restreindre le domaine d'utilisation du prédicat *switchOff* à l'objet *Stove*, par conséquent, le prédicat *switchOff* ne peut être appliqué qu'à un objet *Stove*.

```
(1)<rdfs :Class rdf :ID="#Control_System">
(2)</rdfs :Class>
(3)<rdf :Property rdf :ID="#switchOff">
(4)<rdfs :domain rdf :resource="# Home_Control_System"/>
(5)<rdfs :range rdf :resource="#Stove"/>
(6)</rdf :Property>
```

TABLE 3.3 – Sérialisation d'informations en RDF-S

3.4.3 Les inférences en RDF(S)

Bien que le langage RDF-S permet de décrire la sémantique des informations, il présente cependant certains inconvénients liés à son manque d'expressivité. Il est ainsi impossible : i) de spécifier des contraintes de cardinalité sur une propriété, c'est-à-dire, le nombre de valeurs que peut prendre une propriété. Par exemple, il n'existe aucun moyen d'exprimer qu'une chaise ne peut être occupée que par une seule personne à la fois ; ii) d'exprimer la négation d'une classe ou la disjonction entre deux classes. En RDF-S, il n'est ainsi pas possible d'exprimer qu'une personne n'a pas pris son médicament, ou bien qu'un *Robot* et un *Humain* sont distincts, et iii) d'attribuer une valeur par défaut à un attribut. Les inférences en RDF reviennent à vérifier des relations de type généralisation/spécialisation entre les classes et entre les propriétés. Dans ce cadre, plusieurs langages d'interrogation de documents RDF ont été développés tels que RDQL ((RDF Data Query Language) [12]

12. http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/

et SPARQL [13] (SPARQL Protocol And Rdf Query Language). La particularité de SPARQL est qu'il permet d'extraire des données d'un document RDF saturé. L'opération de saturation consiste à générer tous les triplets RDF satisfaisant des relations de généralisation/spécialisation.

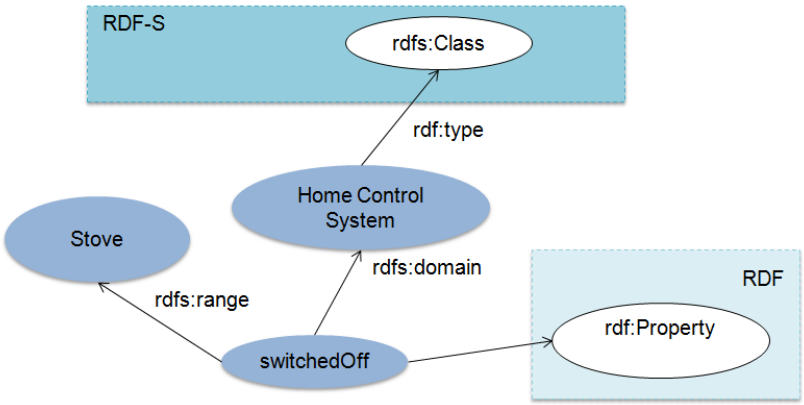

FIGURE 3.6 – Représentation graphique en RDF-S de l'énoncé : *Le système éteint la cuisinière*

3.4.4 Le langage OWL

Pour répondre aux problèmes soulevés ci-dessus, la communauté du Web sémantique a défini un nouveau langage basé sur RDF, mais plus expressif que ce dernier. Il s'agit du langage OWL.

OWL est un formalisme standard pour la représentation des ontologies pour le Web sémantique. Il permet non seulement de représenter les connaissances mais aussi de réaliser des inférences sur ces dernières. Dans le langage OWL, un individu correspond à un objet, une propriété désigne une relation binaire entre objets et une classe représente un ensemble d'objets. Historiquement, le langage OWL est le résultat de la fusion et de l'extension des langages DAML et OIL [75]. Comparé à RDF-S, OWL reprend la syntaxe du langage RDF-S et offre un vocabulaire plus riche pour décrire les propriétés, les classes et les relations entre classes. Le langage OWL comporte aussi un ensemble de constructeurs permettant de décrire des classes complexes, comme la conjonction de classes, ou le quantificateur existentiel.

Dans le langage OWL, on distingue deux types de propriétés binaires : *ObjectProperty* et *DataTypeProperty*. La propriété *ObjectProperty* permet d'associer une classe (représentée avec le constructeur rdfs :domain) aux individus d'une autre

13. http://www.w3.org/TR/rdf-sparql-query/

classe (représentée avec le constructeur rdfs :range). La propriété *DataTypeProperty* permet d'associer une classe à un type de données. Dans les types de données, on trouve les types de bases (entier, flottant, etc.) et les types de données définis dans XML Schéma.

Enfin, il est possible d'associer plusieurs caractéristiques aux propriétés OWL. Nous citons ici deux exemples :

- La propriété transitive (*OWL Transitive Property*) : S'il existe une relation entre les individus x et y et une relation entre les individus y et z, alors il existe nécessairement une relation entre les individus x et z ;
- La propriété fonctionnelle (*OWL Functional Property*) : Il s'agit d'une propriété ne pouvant avoir qu'une seule instance pour un sujet donné. En d'autres termes, s'il existe une relation entre les individus x et y et une relation entre les individus x et z, alors $(y == z)$; y et z représentent le même individu ;

Le langage OWL consiste en un ensemble d'axiomes et de faits pour décrire un domaine [56]. La représentation des connaissances est basée sur la logique de description $\mathcal{SROIQ}$. Il existe de nombreux langages de requêtes basés sur RDF, permettant l'extraction des connaissances ontologiques OWL. Le langage le plus utilisé est le langage SPARQL, qui est devenu une recommandation officielle du W3C [14]. Le langage SPARQL permet d'exprimer une requête sous la forme d'un graphe RDF (les variables de la requête peuvent être des nœuds ou des arcs du graphe). La réponse à une requête consiste à mettre en correspondance cette requête avec un autre graphe représentant la base de connaissances composée de descriptions RDF. Contrairement aux langages de requêtes basés sur RDF, nRQL (new Racer Query Langage) [42] est un langage de requêtes, doté d'une syntaxe propre, et basé sur les logiques de description. Ce langage permet d'extraire des connaissances représentées en OWL-DL et de réaliser des inférences à partir d'une base de connaissances en logique de description (TBox/ABox). Le langage de requêtes OWL-QL [39] est un autre exemple de langage de requêtes basé sur OWL. OWL-QL est un langage formel permettant de définir des requêtes qui peuvent être appariées avec des annotations dans des bases de connaissances et des motifs de réponse. Le langage OWL-QL a été proposé pour la standardisation par le W3C pour être utilisé comme langage et protocole d'interrogation de bases de connaissances web sémantique.

3.4.5 Raisonnement dans OWL

Dans une ontologie OWL, on distingue trois niveaux de raisonnement :

- Le niveau classe : Il s'agit principalement de vérifier l'héritage des classes (subsomption) et de construire la taxonomie des classes de l'ontologie ;
- Le niveau propriétés : Il s'agit de vérifier les relations entre individus de classes différentes à travers différents types de propriétés (transitive, fonctionnelle, etc.). Par exemple, la propriété transitive permet d'inférer l'existence d'une

14. http://www.w3.org/blog/SW/2008/01/15/sparql_is_a_recommendation/

relation entre les individus de deux classes qui n'ont pas de relation explicite dans l'ontologie ;
- Niveau individu : Dans ce niveau, il s'agit d'une part, de vérifier la consistance des connaissances (Consistency Checking) vis-à-vis du modèle d'ontologie, c'est-à-dire de vérifier si une classe possède un individu (instance) dans la base de connaissances, et d'autre part, de rechercher dans la base de connaissances les classes qui correspondent à une description partielle ou complète d'un individu donné ;

L'utilisation des règles permet d'augmenter l'expressivité et les possibilités de raisonnement sur les connaissances exprimées avec les ontologies OWL [55]. L'utilisation des moteurs d'inférence basés sur les logiques de description permet : i) de contrôler la consistance des concepts en se basant sur le principe de subsomption, ii) d'imposer des restrictions sur les cardinalités des classes/propriétés définies au niveau terminologique, et iii) d'effectuer des inférences à partir des instances et des axiomes définis respectivement dans les composants ABox et TBox.

Le langage de règles le plus connu est sans doute SWRL [57]. Ce dernier, considéré comme une extension d'OWL DL, est basé sur la programmation logique, et plus particulièrement sur les clauses de Horn. Il permet de manipuler les variables et les instances déclarées dans le corps des règles. Le principe d'une règle SWRL est de satisfaire l'antécédent de la règle, sans créer un nouveau concept ou une nouvelle relation.

Actuellement, l'indécidabilité du langage SWRL est reconnue comme un obstacle pour combiner les langages OWL et SWRL [64]. Une solution à ce problème consisterait à utiliser des moteurs de règles basés sur l'algorithme Rete [87], comme Jess [15], et CLIPS-OWL [88].

3.5 Utilisation des ontologies en intelligence ambiante et en robotique

L'un des défis importants des systèmes à intelligence ambiante et de la robotique ubiquitaire, est de permettre à des agents de communiquer et de partager une compréhension commune des entités de l'environnement et de leurs interactions. L'autre motivation est de permettre la réutilisation d'un modèle existant dans la construction d'un nouveau modèle par un simple ajout des références aux entités modélisées dans le modèle initial. Ce principe est couramment utilisé pour définir de nouvelles ontologies par importation d'ontologies existantes.

Comme en témoignent plusieurs travaux [157] [18] [138] [12] [67], les ontologies pour le web sémantique constituent aujourd'hui une solution incontournable à

15. http://herzberg.ca.sandia.gov/

la représentation, au raisonnement et au partage des connaissances de sens commun dans le domaine de l'informatique ubiquitaire et de l'intelligence ambiante, notamment pour la mise en œuvre de modèles sémantiques de description de services, des connaissances contextuelles et des architectures des applications.

3.5.1 Raisonnement selon les hypothèses du monde fermé et du monde ouvert

Dans le cadre des systèmes à intelligence ambiante, l'environnement est dynamique, et par conséquent, les formalismes de la logique du premier ordre et les ontologies ne permettent pas de représenter des faits dont la valeur de vérité n'est pas connue à l'avance ou risque de changer continuellement.

Pour traiter ce problème et permettre ainsi une prise de décision du système, il existe deux hypothèses de raisonnement possibles : L'hypothèse du monde fermé (en Anglais : Closed World Assumption- CWA) et l'hypothèse du monde ouvert (en Anglais : Open World Assumption- OWA). Ces deux hypothèses sont toutes les deux utiles pour effectuer des raisonnements dans le contexte d'applications en intelligence ambiante. L'hypothèse du monde ouvert, sur laquelle s'appuie le raisonnement à base d'ontologies, qu'on appelle interprétation logique, considère que l'absence d'un fait dans la base de connaissances présuppose que ce fait est inconnu et non qu'il soit faux. En effet, dans de nombreuses applications, le choix entre les hypothèses CWA et OWA est très important et doit être pris en considération. Dans [102], les auteurs ont étudié la possibilité d'utiliser le raisonnement sur les ontologies pour automatiser les tâches médicales, telles que la sélection cohorte de patients pour des essais médicaux, la surveillance des maladies infectieuses et l'aide à la décision clinique. L'idée fondamentale mise en avant par les auteurs a été de formuler sémantiquement les données de tous les patients participant aux essais médicaux. Les auteurs ont conclu que dans le domaine médical : i) l'hypothèse OWA est utile pour ne pas faire des hypothèses arbitraires sur les résultats si aucune information n'est disponible dans la base de données du laboratoire, et ii) l'assertion qui consiste à affirmer qu'un patient n'est pas sous un traitement par le fait de ne pas être dans la base de données pharmaceutique ne peut être réalisée que sous l'hypothèse CWA.

En général, pour exprimer la négation d'un fait représenté dans une logique de description basée sur l'hypothèse du monde ouvert, il est nécessaire de coder explicitement la négation en introduisant un nouveau rôle dans le composant TBox. Considérons par exemple, le rôle *présent* pour décrire l'énoncé suivant : "présence d'une personne à un endroit donné". La négation de cet énoncé revient à exprimer un nouvel énoncé en introduisant un nouveau rôle comme par exemple *nonPrésent* ou *estAbsent*. Dans un système logique du premier ordre basé sur l'hypothèse du monde fermé, il aurait simplement suffit d'associer l'opérateur de négation au rôle *présent* pour exprimer sa négation.

3.5.2 Représentation et raisonnement sur l'affordance

L'affordance est un concept introduit par Gibson [44] pour désigner les relations existant naturellement entre un individu et son environnement.

En robotique, l'affordance permet de modéliser sémantiquement comment un agent peut interagir avec les objets de son environnement ou ce qu'il faut faire avec ces objets [53]. Cette modélisation permet de coupler la perception de l'environnement et l'action du robot. La modélisation de l'environnement concerne trois niveaux : (i) Le niveau bas où l'information est encodée en XML ; (ii) le niveau intermédiaire qui permet de représenter les états du monde sous forme d'individus. Autrement dit, chaque individu représente un objet réel tel qu'un robot, un réfrigérateur, un obstacle, etc. Chaque individu est stocké dans le composant ABox ; (iii) le niveau haut décrit l'ontologie de concepts correspondant au composant TBox.

Galindo et al. [41] proposent un modèle multi-hiérarchique incluant une hiérarchie

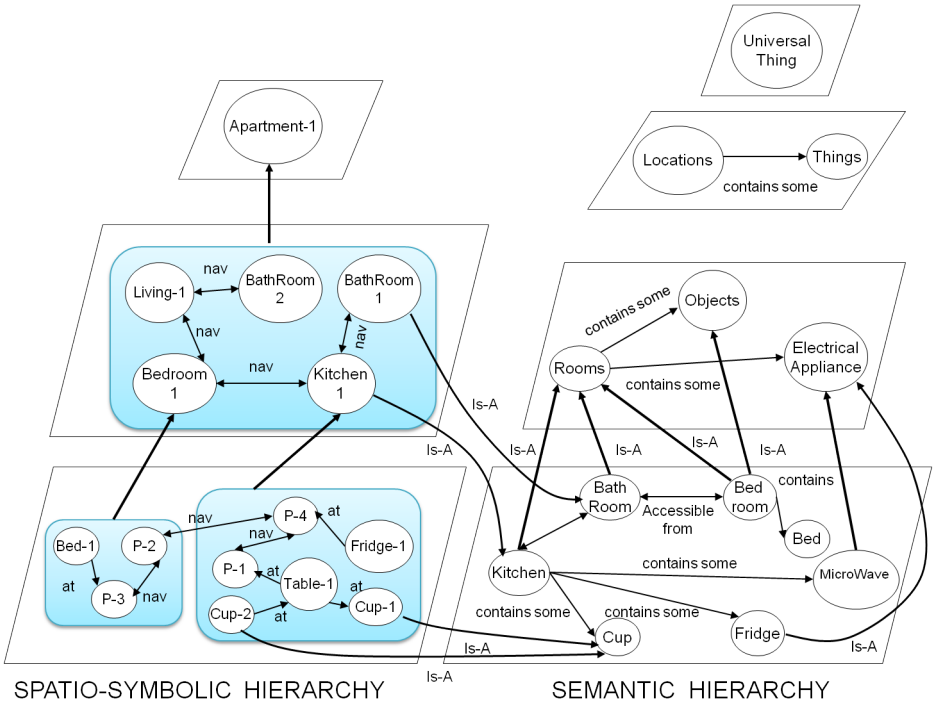

FIGURE 3.7 – Hiérarchie spatiale et sémantique [41].

représentant des symboles (Room, Kitchen, Table, etc.) dont certains sont associés à des informations spatiales, figure 3.7. Les symboles sont représentés par des sommets et les relations entre symboles par des arcs de type :

- Navigability (navigabilité) : Cette relation exprime la possibilité d'accéder de l'espace *Bedroom 1* vers l'espace *Bathroom* ;
- Located-at (situé à) : Cette relation exprime une relation de proximité comme par exemple, *Cup-1* se trouve sur *Table-1* ;

Les sommets de la hiérarchie spatiale sont abstraits en niveaux supérieurs de la hiérarchie pour la représentation topologique de l'environnement qui est en général une représentation moins détaillée. La seconde hiérarchie, représente l'information sémantique décrivant des relations de type (*sorte-de et est-un*) entre les concepts. Il s'agit ici de modéliser des catégories de symboles spatiaux et des relations entre ces symboles à l'aide d'un réseau sémantique en utilisant le langage NeoClassic [103]. Ce dernier est basé sur le langage de description d'ontologies CLASSIC [104].

3.5.3 Sensibilité au contexte

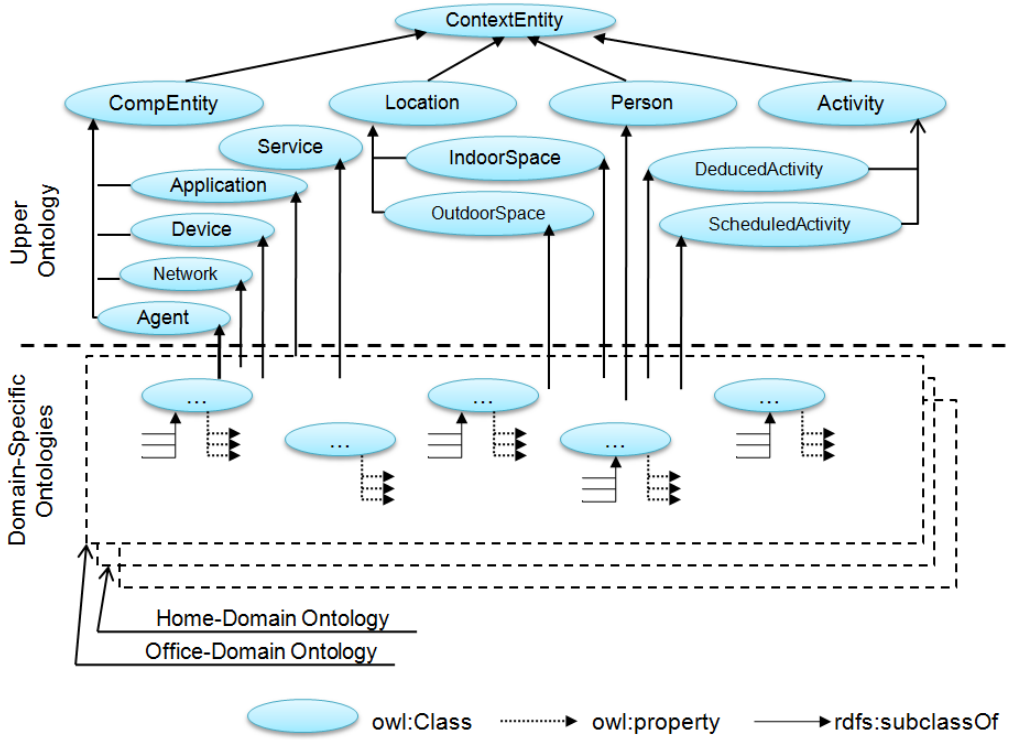

FIGURE 3.8 – Fragment de l'ontologie CONON [149]

L'utilisation du web Sémantique pour le développement de plateformes de gestion sémantique des connaissances contextuelles en intelligence ambiante a fait objet de plusieurs travaux de recherches qui ont débouché sur plusieurs ontologies, telles que, CONON [149], SOUPA [30], BeAware [174], SAWA [155], etc. Le point commun entre ces ontologies réside, d'une part, dans l'utilisation des concepts et des propriétés du langage d'ontologie OWL pour décrire les informations contextuelles caractérisant un contexte et d'autre part, l'utilisation d'un langage de règles web sémantique pour inférer de relations entre ces informations pour la reconnaissance du contexte. L'ontologie CONON (CONtext ONtology), développée dans le cadre du projet SOCAM [168], est sans doute l'une des ontologies les plus représentatives, qui ont influencé nos recherches pour la mise en œuvre d'un

mécanisme de raisonnement réactif pour l'adaptation au contexte.

Dans l'ontologie CONON, la modélisation du contexte est centrée sur la notion d'application. Les concepts sont définis selon le domaine considéré. Ce dernier correspond à l'environnement dans lequel s'exécutent les applications, comme par exemple, la maison, le bureau, la voiture, etc. Afin de garantir les propriétés de généricité et d'extensibilité du modèle du contexte, l'ontologie CONON a été divisée en deux ontologies principales. La première ontologie, appelée "Generalized Ontology", permet de décrire des concepts généraux du contexte comme par exemple les activités : préparer un repas, regarder la télévision, etc. La deuxième ontologie, appelée "Domain Specific Ontology" permet, quant à elle, de relier les concepts génériques avec les concepts du domaine pour une application spécifique, figure 3.8. Cette décomposition a pour objectif de réduire la complexité de l'ontologie du contexte pour ne permettre que le traitement des connaissances relatives au domaine d'application. Chaque domaine décrit des objets physiques ou conceptuels où chaque concept peut être associé, soit à un attribut de contexte grâce à la relation *owl :DatatypePropery*, soit à d'autres concepts, à l'aide de la relation *owl :ObjectProperty*. Le raisonnement dans CONON s'effectue en deux étapes : La première consiste à effectuer des inférences en logique de description pour vérifier la satisfaisabilité des concepts (voir paragraphe 3.3.6) et la relation de subsomption entre les concepts. La seconde étape utilise, quant à elle, des clauses de Horn pour la reconnaissance du contexte. À titre d'exemple, pour inférer qu'une personne est en train de dormir, les auteurs définissent les deux règles suivantes :

règle en logique de description :

$$(?rdf : type \;\; owl : TransitiveProperty) \wedge (?A \;\; ?P \;\; ?B) \wedge \;\; (?B \;\; ?P \;\; ?C) \\ \Longrightarrow ((?A \;\; ?P \;\; ?C) \tag{3.1}$$

Où ?A, ?B et ?C sont des concepts/individus et ?P une propriété.

règle en logique du premier ordre :

$$(?u \;\; locatedIn \;\; Bedroom) \wedge (Bedroom \;\; lightLevel \;\; LOW) \wedge \\ (Bedroom \;\; drapeStatus \;\; CLOSED) \\ \Longrightarrow (?u \;\; situation \;\; SLEEPING) \tag{3.2}$$

La règle 3.1 permet de vérifier la propriété de satisfaisabilité et la relation de subsomption entre concepts. La règle 3.2 signifie que si une personne représentée par le symbole *u* est localisée dans une *chambre*, et que le *rideau* de cette *chambre* est fermé, le système déduit alors que la personne *u* dort.

L'utilisation de l'ontologie CONON dans la plateforme SOCAM permet aux applications de s'adapter dynamiquement au contexte de l'environnement

de l'utilisateur, figure 3.9. Ainsi, lorsqu'un utilisateur change d'environnement, l'ontologie du domaine d'application, qui correspond à l'environnement courant, est chargée puis associée dynamiquement à l'ontologie générique du contexte. Ainsi, si l'utilisateur est dans un environnement résidentiel, l'ontologie du domaine résidentiel est associée à l'ontologie générique pour décrire le contexte de l'environnement résidentiel. Les services liés à ce contexte sont alors proposés à l'utilisateur. De même, quand l'utilisateur quitte l'environnement résidentiel pour l'environnement de type *automobile*, l'ontologie du domaine *automobile* se substitue à l'ontologie du domaine résidentiel. Dans ce cas, un nouvel ensemble de services liés au contexte *automobile* est alors proposé à l'utilisateur, comme par exemple, tous les appels téléphoniques sont redirigés vers une boite de messagerie vocale ou vers un téléphone mobile. Dans l'intergiciel SOCAM, le contexte est représenté sous la forme de prédicats du type : *Predicate (subject, value)*, où *Predicate* représente le nom d'un prédicat, tel que *locatedAt* ; *subject* représente un concept, comme par exemple, *Living_Room, Room, Person, etc.* et enfin, *value* correspond à l'état de *subject*.

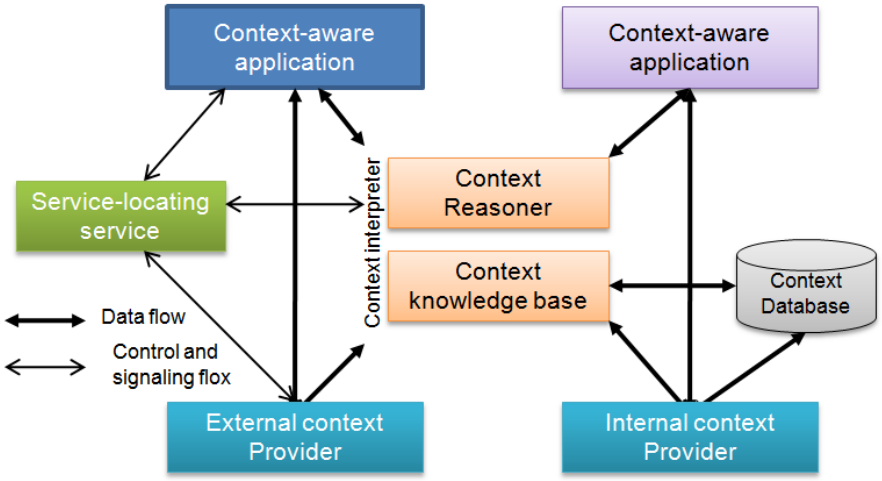

FIGURE 3.9 – Reconnaissance de contexte avec SOCAM [168]

Dans SOCAM, le raisonnement est utilisé soit pour détecter les incohérences dans la base de connaissances ou la reconnaissance de contexte. Les deux règles ci-dessous illustrent des exemples de règles utilisées pour la reconnaissance du contexte :

1. (?user rdf :type socam :Person), (?user, socam :locatedIn, socam :Bedroom), (?user, socam :hasPosture, "LIEDOWN"), (socam :Bedroom, socam :lightLevel, "LOW"),
   (socam :Bedroom, socam :doorStatus, "CLOSED")
   − > (?user socam :status "SLEEPING") ;
2. (?user rdf :type socam :Person), (?user, socam :locatedIn, socam :BathRoom),

socam :WaterHeater, socam :status, "ON"), (socam :BathRoom,
socam :doorStatus, "CLOSED")
$->$ (?user socam :status "SHOWERING") ;

Dans le projet COBRA, *Chen et al.* ont proposé l'ontologie SOUPA pour modéliser le contexte des applications d'intelligence ambiante [29] . Cette ontologie permet de décrire par exemple, le dispositif informatique utilisé par l'utilisateur, le lieu ou l'application est utilisée, etc. SOUPA utilise des concepts génériques importés à partir de plusieurs ontologies de haut niveau telles que : FOAF [16], Rei [17], RCC [169], etc. Ces dernières sont alignées sur l'ontologie universelle OpenCyc. Le raisonnement sur le contexte est basé sur des règles d'inférence écrites dans le langage F-Logic [165].

Dans le projet AMIGO, *Euzenat et al.* proposent une ontologie de contexte permettant de décrire des catégories de contextes. Les auteurs introduisent pour chaque objet physique ou virtuel caractérisant un environnement d'intelligence ambiante, un nouveau concept décrivant son contexte [25]. Par exemple, l'espace chambre est associé au concept "Room Context". Ce dernier est dérivé d'un concept générique appelé *Context*. L'utilisation du raisonnement de la logique de description permet d'inférer des relations entre les instances des concepts contextuels.

D'autres approches utilisent le concept *Situation* pour modéliser le contexte. Rappelons que le contexte est l'ensemble des informations permettant de caractériser, même partiellement, la situation dans laquelle se trouve un objet de l'environnement. Dans le projet Situation AWareness Assistant (SAWA) [155], les auteurs, proposent d'une part, une ontologie OWL pour la représentation des connaissances contextuelles, et d'autre part, des règles d'inférence écrites dans le langage de règles web sémantique SWRL pour la reconnaissance de situations. L'ontologie SAWA ne prend pas en compte la description du temps et de l'espace. *Mastrogiovanni et al.* [153] proposent une autre approche permettant de déduire des relations implicites pouvant exister entre des objets de l'environnement à partir de relations explicites. Par exemple, la propriété *inBed* décrit une relation entre un utilisateur et un lit. Les auteurs se basent sur les mécanismes de raisonnement de la logique de description. L'inconvénient de l'ontologie SAWA est liée à l'utilisation de relations spécifiques pour chaque type d'objet, ce qui pose un problème de généricité du modèle du contexte. Enfin, dans le projet BeAware, les auteurs proposent une ontologie générique pour la description de situations. Cette ontologie est centrée sur les concepts : Objet, événement et relation. Le concept relation permet de décrire toutes les relations spatio-temporelles que peuvent avoir les objets du monde réel observés dans le cadre d'une situation.

16. http://www.foaf-project.org/
17. http://www.csee.umbc.edu/~lkagal1/rei/

3.6 Utilisation des ontologies dans les systèmes robotiques

L'utilisation avec succès des robots dans les domaines industriels a poussé les chercheurs à les intégrer dans le quotidien des humains (services de nettoyage, de surveillance, etc.). Le concept de robot companion, proposé récemment par les roboticiens, témoigne de cette volonté de développer des entités physiques ou virtuelles pour aider les personnes dans leur vie quotidienne. À ce concept est souvent associé le concept plus général de la robotique ubiquitaire qui vise à offrir à des utilisateurs divers services en tout lieu et à tout moment [59]. L'intégration des robots dans les environnements à intelligence ambiante constitue un challenge important qui commence à faire l'objet de nombreux travaux de recherche à travers le monde. Ces robots sont destinés à assurer des tâches complexes comme l'assistance cognitive et la surveillance des personnes âgées dépendantes. Ces tâches nécessitent de manipuler des connaissances concernant les propriétés des objets et l'exécution des tâches et des actions requises. Pour ce faire, les robots doivent être capables de s'adapter à des environnements dynamiques, c'est à dire, de traiter, d'analyser, de corréler et de comprendre les événements, des situations dont l'être humain est l'acteur principal. Plusieurs travaux récents ont abordé l'utilisation des ontologies pour mettre en œuvre des plateformes de robotique ubiquitaire offrant des fonctions de perception augmentée du contexte ambiant [96] [97] [140] [161] [73] [107] et des fonctions de planification de tâches et de composition de services robotiques dans des environnements à intelligence ambiante [156] [121] [158].

Les travaux sur la représentation symbolique sémantique des connaissances et les raisonnements associés est une problématique clé de la robotique [84]. Les premiers travaux ont consisté à proposer des approches basées sur la logique comme langage universel de la représentation des connaissances manipulées par le robot. Dans [23], les auteurs recensent les premiers travaux réalisés dans ce cadre. *Shakey* a été le premier projet de robot autonome utilisant une base de connaissances symboliques pour la planification de tâches.

Les avancés réalisées dans le domaine de la représentation sémantique des connaissances grâce aux apports du web sémantique, ont encouragé la communauté robotique à s'intéresser aux approches sémantiques, notamment, l'utilisation des ontologies pour représenter les connaissances du robot. Ces travaux sont basés sur le processus d'ancrage des ontologies à partir des entités observées dans le monde réel. Par ce processus, un concept dans une ontologie devient reconnaissable à partir des données remontées par les capteurs. L'ancrage d'ontologies "Ontology grounding" est particulièrement bien adapté pour des problématiques où des données non-structurées sont disponibles en masse, mais qui doivent être converties manuellement en métadonnées structurées [127]. Le processus d'ancrage s'appuie sur des techniques d'apprentissage permettant la classification d'instances de

concepts, mais requiert également l'utilisation de techniques d'extraction des connaissances telles que SPARQL.

En robotique, le problème d'ancrage des symboles d'une ontologie est traité dans diverses applications. Par exemple, dans le cas d'un agent incarné, toutes les opérations sensori-motrices correspondant à l'interaction de ce dernier avec son environnement sont ancrées sous formes de symboles dans sa base de connaissances. On peut citer dans ce cadre, les travaux de *Lemaignon* et *Alami* [70], ou ceux de *Tenorth et Beetz* [145] sur l'utilisation de la plateforme Knowrob dans le cadre d'applications impliquant des interactions humains-robots de service.

Dans le cas d'un système multi-robots, l'ancrage des symboles est complexifié par le fait que les symboles doivent être partagés entre plusieurs robots dont les représentations de connaissances sont souvent hétérogènes [172]. On parle dans ce cas d'ancrage social [127]. La plateforme RobotEarth est un exemple d'application utilisant ce principe [148]. Enfin, dans le cas des applications robotiques basées sur des agents non incarnés, l'ancrage des perceptions du robot permet de créer un lien entre les perceptions de bas niveau, comme par exemple, la vision ou la localisation, avec des représentations sémantiques de haut niveau [171].

Dans ce qui suit, nous présentons une synthèse des travaux sur l'utilisation des ontologies dans les systèmes robotiques. L'objectif communément visé est d'ancrer et de partager les connaissances symboliques dans le monde physique du robot à l'aide des symboles d'une ontologie et d'effectuer des raisonnements en ligne sur ces symboles.

3.6.1 La plateforme ORO

ORO est une plateforme de gestion de connaissances symboliques permettant la mise en œuvre d'applications en robotique de service impliquant des interactions humain-robot [70]. Les auteurs introduisent la notion d'évènements sémantiques dans les différents niveaux de contrôle du robot pour aboutir à une programmation des comportements de ce dernier, qui soit réactive expressive et abstraite des contingences de bas-niveau. Dans cette approche, déclencher par exemple un comportement spécifique du robot lorsqu'un humain observe le robot tout en étant assis s'exprime littéralement comme suit :
([* type Human, * looksAtmyself, *isSitting true], behaviour_callback()).

La plateforme ORO est basée sur l'ontologie "OpenRobots Common SenseOntology" écrite en OWL [70], figure 3.10. Cette ontologie permet de construire une base de connaissances ontologiques dites de sens commun dont le contenu est extrait à partir du web. L'ontologie ORO est générique et facilement extensible, étant donné qu'elle est alignée sur l'ontologie de haut niveau OpenCyc. Elle comprend plus de 56 classes permettant de décrire sémantiquement les connaissances

concernant la perception et l'interaction du robot, dans le cadre d'applications en robotique de service. Le choix du langage OWL présente l'avantage de permettre l'échange et l'ajout des connaissances dans l'ontologie OpenRobots à partir d'autres sources terminologiques fournies par d'autres robots ou par des bases linguistiques standards telles qu'OpenMind et WordNet.

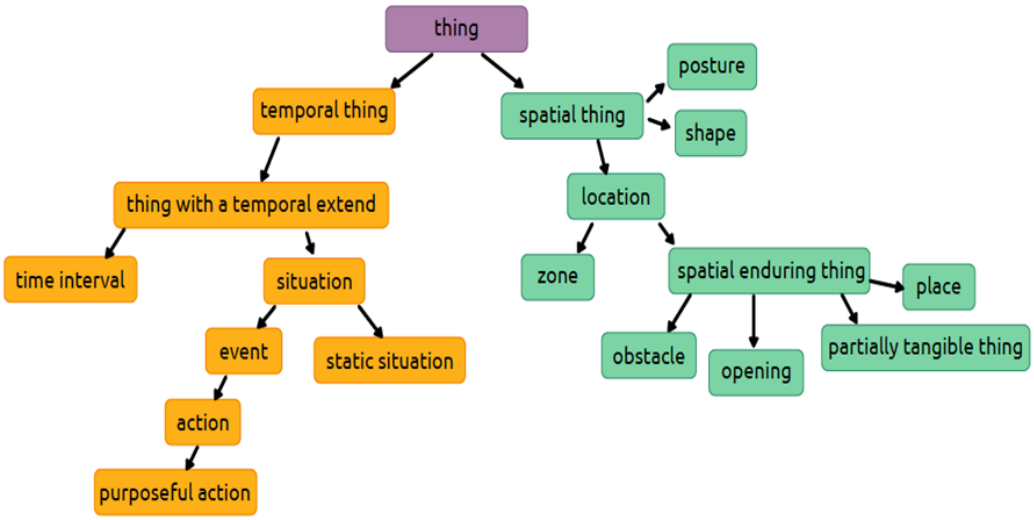

FIGURE 3.10 – Concepts de haut niveau de l'ontologie ORO [70].

L'ontologie ORO est gérée à travers un serveur offrant une interface d'accès distant générique et simple. Cette interface permet de gérer les opérations comme l'ajout, la rétractation, la mise à jour de faits symboliques, ainsi que l'accès à ces derniers par plusieurs mécanismes de requêtes, tout en garantissant des temps de réponse acceptables par rapport au contrôle des tâches du robot. La plateforme ORO intègre aussi des outils de parcours taxonomique de l'ontologie permettant de lier les différentes classes et propriétés par leurs liens de subsomption et d'instanciation. Pour ce faire, les deux moteurs d'inférences Jena et Pellet sont utilisés conjointement pour la gestion des ajouts et des suppressions de connaissances.

L'autre avantage de la plateforme ORO est qu'elle permet de maintenir plusieurs modèles symboliques en parallèle. Chaque modèle est indépendant et cohérent d'un point de vue logique, ce qui permet d'effectuer des raisonnements en utilisant des vues cognitives différentes sur l'environnement. Ainsi, un objet peut être visible pour le *robot*, mais invisible pour l'homme. Les observations (visible et invisible) sont exprimées à l'aide des propriétés $isVisible true$ et $isVisible false$, définies dans deux modèles différents.

La plateforme ORO intègre aussi un module externe au serveur ORO appelé

SPARK (SPAtial Reasoning & Knowledge). Ce dernier permet d'effectuer des raisonnements pour inférer des faits symboliques de nature géographique issus de l'analyse visuelle de l'environnement du robot. Cependant, la représentation des connaissances temporelles n'est pas prise en charge dans la version actuelle d'ORO. Toutes les instances OWL-DL stockées dans la base de connaissances du robot représentent les croyances courantes du robot.

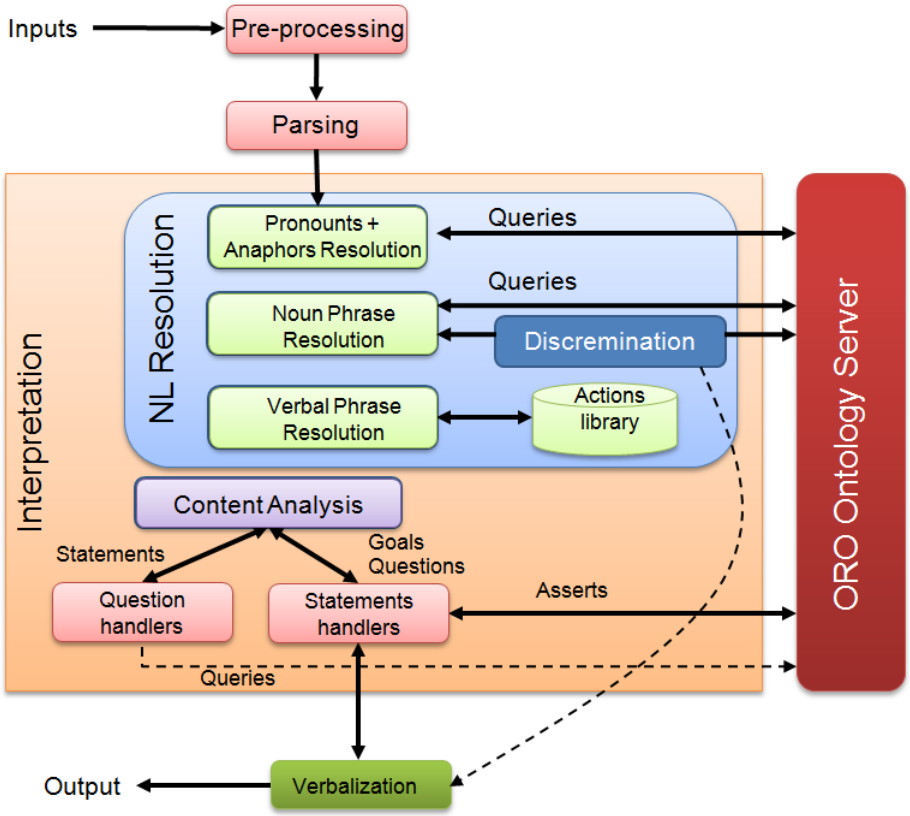

FIGURE 3.11 – Les modules de l'architecture DIALOGS [70].

La plateforme ORO intègre un système appelé Dialogs, permettant de traiter la sémantique des termes utilisés en langage naturel dans les interactions homme-robot. Ce système est ainsi utilisé pour permettre à une personne d'ordonner au robot d'exécuter des tâches de manipulation d'objets en lui indiquant une série de phrases contenant des mots clés spécifiques, figure 3.11. Dialogs analyse grammaticalement et sémantiquement des énoncés simples exprimés en Anglais. Il propose par ailleurs une interprétation de ces énoncés et convertit le cas échéant la phrase initiale en une série de nouveaux faits symboliques. Dialogs inclut des stratégies interactives pour éliminer les ambiguïtés dans des phrases. Le traitement du dialogue utilise l'ontologie ORO et permet surtout de fournir le modèle de

connaissances nécessaire à l'ancrage sémantique des phrases. Ainsi, une phrase comme "Apporte-moi le livre" prête à confusion dès lors que deux livres sont présents dans la scène.

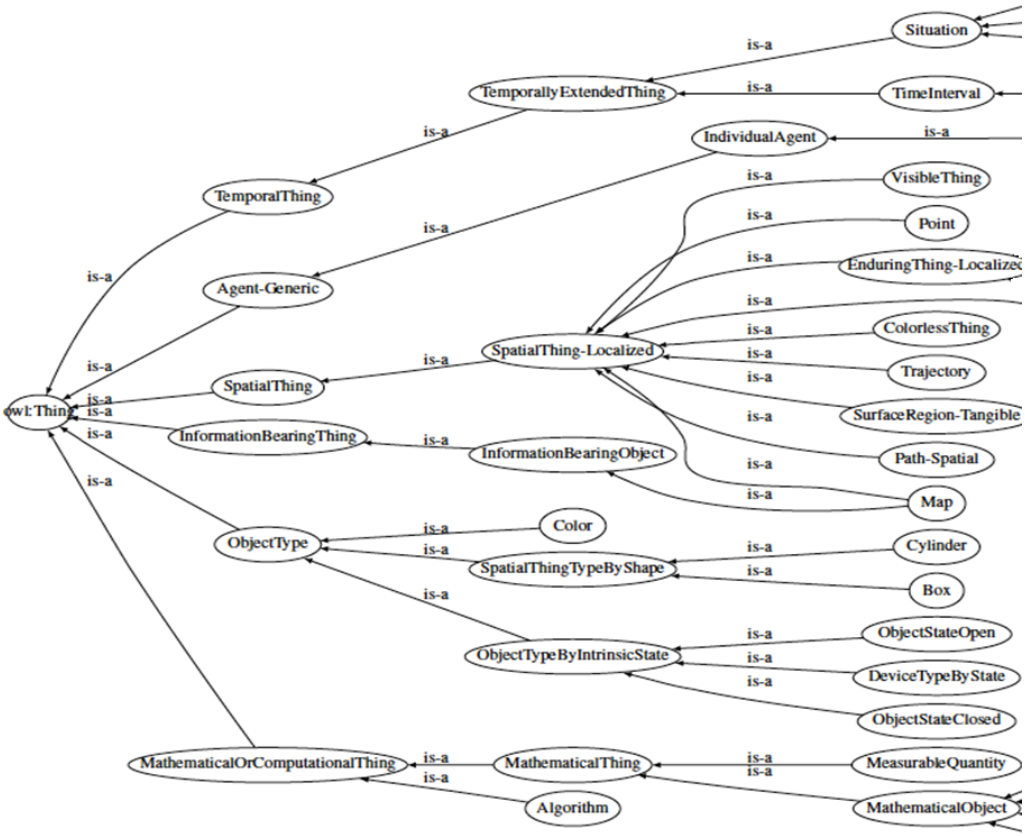

FIGURE 3.12 – Principaux concepts de haut niveau de l'ontologie KnowRob [145].

3.6.2 KnowRob

KnowRob [145] est un système de gestion de connaissances symboliques, développé par l'université technique de Munich (TUM). L'ontologie KnowRob est alignée sur l'ontologie OpenCyc. Les branches les plus importantes sont représentées par les classes TemporalThing, TimeInterval et SpatialThing, et décrivent respectivement des événements, des actions et objets de représentation spatiaux tels que des points ou des trajectoires. La modélisation des événements dans l'ontologie KnowRob est similaire à celle proposée dans OpenCyc, figure 3.12.

Les connaissances factuelles sont manipulées à l'aide du langage Prolog et sont associées à des prédicats spécifiques appelés prédicats calculables (*computable predicate*). Ces derniers sont calculés en appelant des méthodes externes au module Knowrob pour évaluer, par exemple, les relations spatiales du type *"au dessus d'un*

objet" ou *"à l'intérieur d'un objet"*. Cette manière de faire évite de recourir à des procédures complexes de réification de triplets RDF et de vérification de relations d'héritage entre classes. Les prédicats calculables permettent de générer des symboles correspondant à la perception du monde, et de mettre à jour les croyances du robot suite à un changement des observations. Know-Rob permet ainsi au robot de s'assurer que n'importe quel symbole dans la base de connaissances est lié à des structures de données qui sont traitées par les modules de perception et de contrôle du robot.

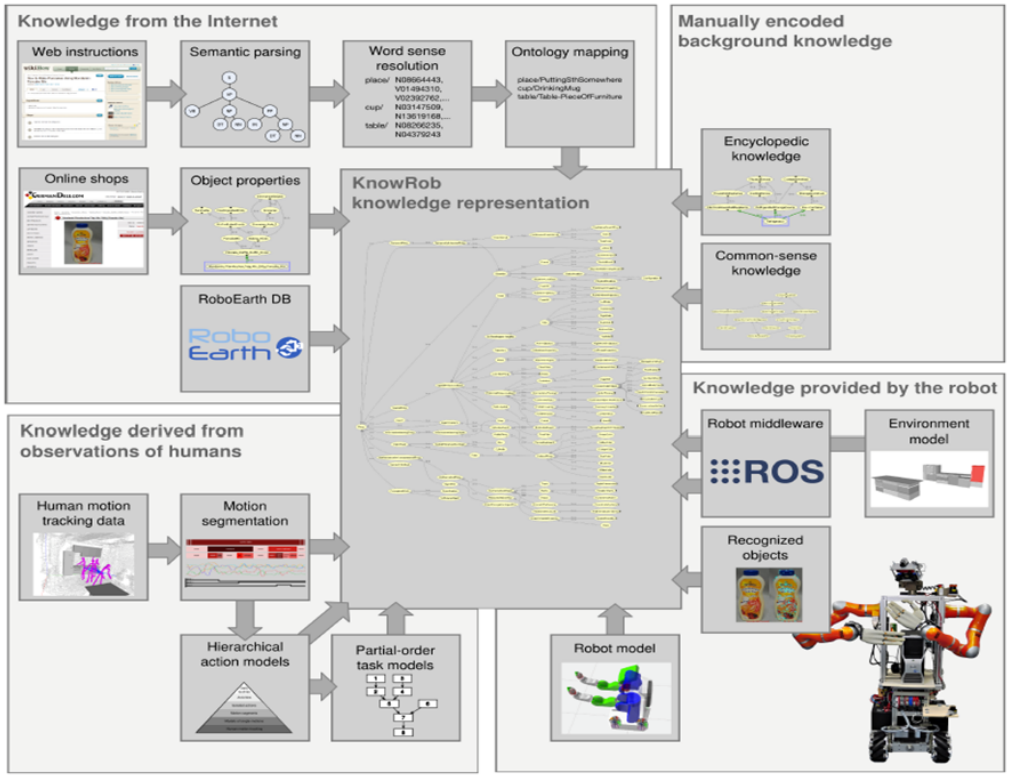

FIGURE 3.13 – Architecture CRAM [15].

Know-Rob a été utilisée par *Beetz* [15] pour la mise en œuvre de CRAM (Cognitive Robot Abstract Machine), une plateforme de planification de tâches de robots autonomes. De nombreux scénarii ont été considérés comme par exemple chercher des objets cachés puis les déplacer, ou encore préparer des repas à partir de menus extraits de la base internet *Wikihow*. La planification des tâches du robot est basée sur le langage CRAM PLAN Language (CPL). Ce langage expressif permet de spécifier des comportements réactifs du robot qui sont guidés par ses capacités sensoriels. Un plan CPL définit comment le robot répond aux événements issus des capteurs, et aux changements dans les croyances du robot. Ainsi, le langage CPL permet non seulement de superviser l'exécution des plans mais aussi

de raisonner sur la manière dont le plan a été exécuté pour comprendre la cause de l'échec d'un plan et de détecter si les croyances dont le robot dispose sont erronées. Il faut noter que les croyances du robot sont mises à jour à partir du résultat des actions exécutées. L'ensemble de ces croyances est extrait à partir de la plateforme Know-Rob, figure 3.13.

Know-Rob et CRAM sont les modules centraux de la plateforme d'interconnexion de robots au web à large échelle RobotEarth. Dans [148], les auteurs présentent robotEarth comme une plateforme web open source dont l'objectif est de permettre à un robot de partager des connaissances sur sa perception de l'environnement et sur les actions qu'il doit exécuter en collaboration avec des humains ou avec d'autres robots. Les connaissances échangées dans RoboEarth sont représentées sous forme d'annotations OWL selon le format préconisé par l'initiative *LinkedData*. Ainsi, quand le robot échange des connaissances à travers la plateforme RobotEarth, il doit être capable de sélectionner uniquement les connaissances qui sont utiles pour son contexte courant. L'architecture de RobotEarth est composée de trois couches principales : La couche centrale consiste en un serveur de base de données pour stocker les données relatives aux objets, cartes de navigation et tâches à exécuter. La deuxième couche concerne la modélisation et la cartographie sémantique de l'environnement, la reconnaissance et l'apprentissage de situations et l'exécution de tâches. La troisième couche permet une abstraction des fonctions spécifiques de chaque robot.

L'un des scénarii mis œuvre avec Know-Rob [145] est celui d'un robot assistant un patient à l'hôpital. Dans ce scénario, le robot doit d'abord chercher une bouteille contenant une boisson ; ce qui nécessite du robot de se diriger d'abord vers le placard où se trouve la bouteille, de prendre cette dernière puis, de se diriger vers le patient allongé sur son lit. Lors de l'exécution de ces tâches, le robot met à jour les positions des objets à chercher dans une base de connaissances commune. Cette dernière est ensuite utilisée par les autres robots pour l'exécution d'autres tâches. Ce scénario, a priori simple, montre les défis à résoudre en termes de stockage, de réutilisation et de partage de connaissances pour la planification et l'exécution des tâches des robots. L'idée d'exploiter le web comme source d'information semble a priori séduisante ; cependant, elle pose des problèmes non-triviaux :

- L'hétérogénéité des informations issues de plusieurs sources et celles produites par des humains et des robots. Ceci soulève le problème de représentation d'informations dans un format et un langage communs permettant de décrire formellement une information en incluant sa sémantique. Ceci pose également le problème de l'alignement d'ontologies en supposant qu'à chaque source est associée une ontologie ;
- Le grand volume d'informations disponibles peut ralentir considérablement les performances en temps de calcul des inférences ;

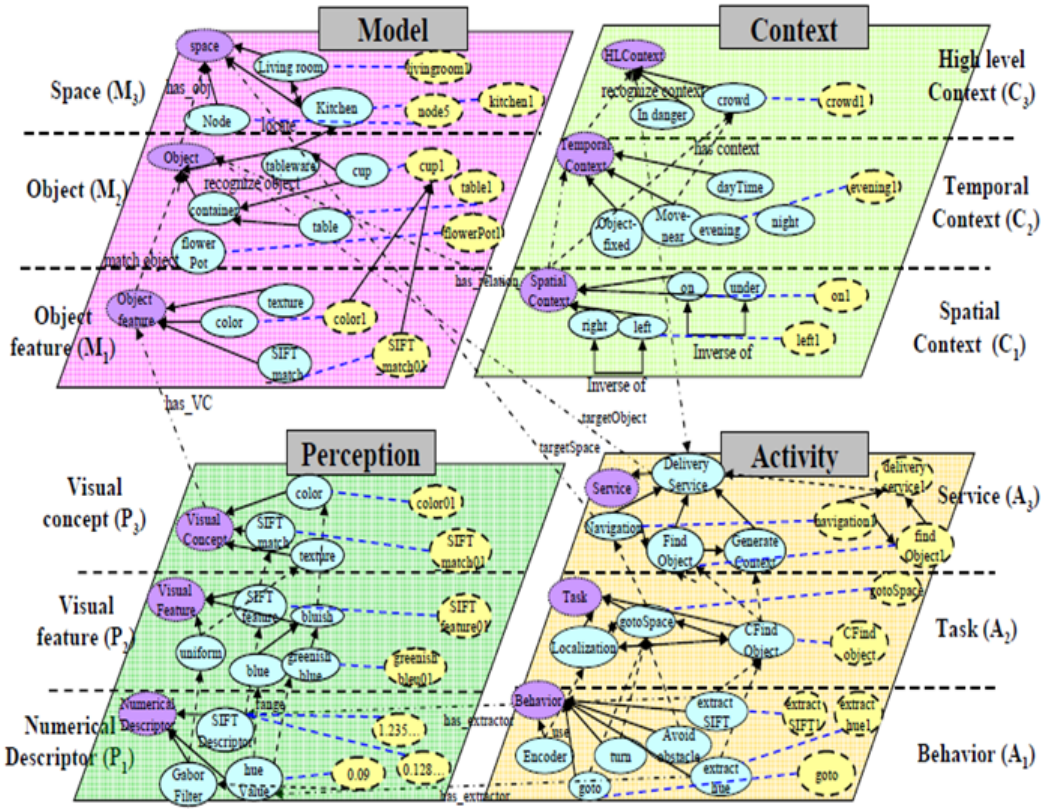

FIGURE 3.14 – L'ontologie OMRKF et ses quatre niveaux de connaissances : Perception, Modèle, Contexte et Activité [170].

3.6.3 La plateforme OMRKF

OMRKF "Ontology-based Multi-layered Robot Knowledge Framework" est une plateforme de représentation et de raisonnement sémantique multidimensionnelle [170]. Elle utilise des clauses de Horn associées à une ontologie OWL-DL décrivant des hiérarchies de concepts. L'ontologie proposée dans OMRKF repose sur quatre couches conceptuelles permettant de décrire respectivement les perceptions, les activités, le contexte et le modèle de l'environnement du robot, figure 3.14. Les couches conceptuelles *Perception, Model et Activity* permettent de modéliser les connaissances nécessaires à la navigation du robot. La couche *Context* permet de modéliser les différents caractéristiques de l'environnement. Chacune de ces couches conceptuelles possède une description exhaustive composée de trois niveaux : Niveau haut, niveau intermédiaire et niveau bas. Par exemple, le niveau haut de la couche conceptuelle *Model* permet de décrire l'espace où se trouve le robot, tel que *Living_Room, Kitchen* ; le niveau intermédiaire décrit les objets se trouvant dans un espace, tel que *Table* ou *Cup*. Le niveau bas permet de décrire les caractéristiques visuelles de ces objets telles que la texture, la couleur, etc.

Bien que le raisonnement soit basé sur les clauses de Horn, les auteurs ne fournissent aucune précision sur la spécification et la manière dont les actions du robot sont planifiées ou mises en œuvre. De notre point de vue, la complexité du modèle OMRKF peut poser d'une part, un problème de performance dû au temps nécessaire pour effectuer tous les appariements d'instances et de concepts des différentes couches, et d'autre part, des problèmes de décidabilité et de passage à l'échelle notamment dans le cas où les appariements des variables dans les règles peuvent déclencher des règles de contrôle conflictuelles pouvant conduire à des connaissances contradictoires.

Le tableau 3.4 résume les propriétés des principales plateformes de représentation et de raisonnement à base d'ontologies pour l'intelligence ambiante et la robotique, que nous avons étudiées dans ce chapitre.

Critères Techniques	ORO	KnowRob	OMRKF	CONON	Beware	SAWA	AMIGO	SOUPA
Utilisation du langage OWL-DL	+	+	+	+	+	+	+	+
Alignement avec des ontologies universelles	OpenCyc	OpenCyc	N/I	+	X	X	X	OpenCyc
Langage d'extraction des connaissances	SPARQL	SPARQL	+	N/I	N/I	OWL Query Language	OWL Query Language	N/I
Langage de règles utilisé	Prolog	Combinaison RDF/Prolog	Clauses de Horn	Clauses de Horn (SWRL)	Clauses de Horn vers Jess	N/I	Clauses de Horn (Flora)	
Utilisation du raisonnement DL	+	+	+/-	+	+	N/I	N/I	+/-
Application du raisonnement	HRI	HRI et Planning	HRI	Supervision AmI	Supervision Trafic Routier	Supervision Logistique	Supervision AmI	Supervision AmI
Présupposition du nom unique (UNA)	X	X	X	X	X	X	X	X
Prise en compte de la négation par l'échec pendant l'inférence	+/- (Prolog)	+/- (Prolog)	.	X	X	+/- (Jess)	X	Flora
Possibilité de modifier et/ou supprimer les instances pendant l'inférence	+/- (Prolog)	+/- (Prolog)	X	X	X	X	X	+/- (Flora)
Abstraction de haut niveau du contexte en utilisant une ontologie générique	+/-	+/-	+	+	+	+	+	+
Représentation spatio-temporelle des évènements dans l'ontologie	X	X	X	X	Situation et Event	X	+/-	Event (DAML-Time)
Raisonnement temporel sur les instants (Estampille temporelle)	+/-	+/-	+/-	X	+	X	X	+
Raisonnement temporel sur les intervalles de temps	X	X	X	X	+	X	X	X
Raisonnement sur des connaissances incomplètes	X	+/-	X	X	X	X	X	X
Légende	**X :** non ; **+ :** Oui ; **N/I :** non indiqué ; **+/- :** Pas explicitement ; **HRI :** Human Robot Interaction							

TABLE 3.4 – Comparaison des principales plateformes pour l'intelligence ambiante et la robotique en termes de représentation et de raisonnement à base d'ontologies

3.7 Représentation et raisonnement sur des connaissances dynamiques

En Intelligence Artificielle, les approches les plus connus sont celles de *Mac Dermott* [86] et d'*Allen* [2]. *McDermott* vise à modéliser deux aspects : l'indétermination du futur et la continuité du temps. La logique chronique de *McDermott* définit un fait comme un ensemble d'états où la valeur de vérité d'une proposition est toujours vraie. Les différents états sont ordonnés par une relations anti-symétrique et transitive noté $<$. Ainsi, le futur selon *McDemrott* est vu comme un arbre avec plusieurs directions possibles. Un des chemins possibles dans cet arbre correspond à une chronique. Contrairement à la logique de *McDermott*, la logique d'*Allen* détaillé dans le paragraphe suivant, représente le temps sur un axe temporel.

3.7.1 La logique temporelle d'Allen

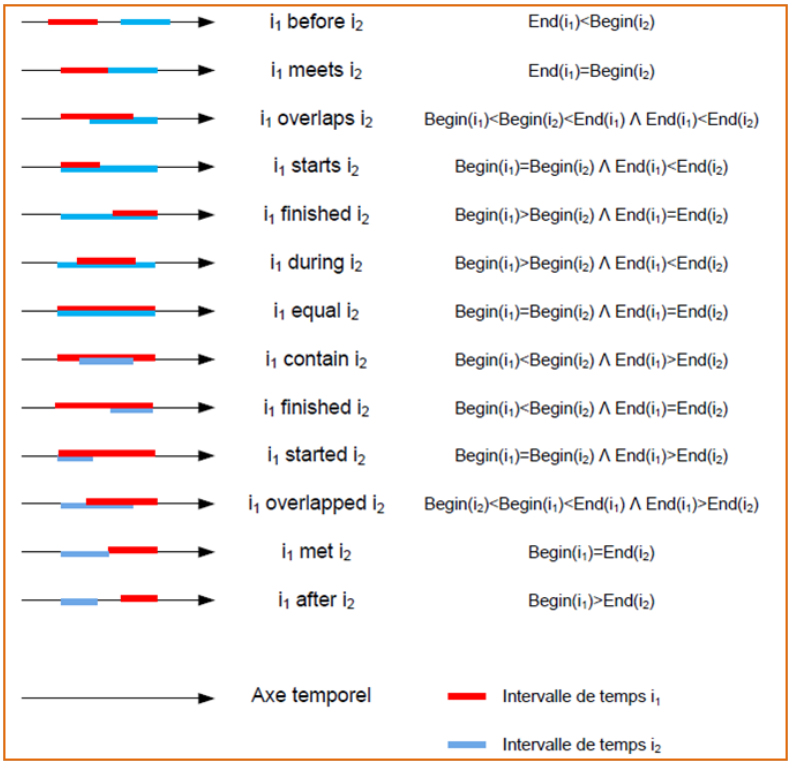

FIGURE 3.15 – Relations sur les intervalles temporels d'Allen [51].

Allen représente la dimension temporelle par des intervalles caractérisé chacun par deux points correspondant à ses extrémités (début et fin) [2] [3] [4]. Il propose treize relations temporelles comprenant 7 relations de base (*equal, before, during, meets, starts, overlaps, et finishes*) et six relations inverses (*after, contain, met, overlapped, finished*), figure 3.15. Le formalisme d'Allen est utilisé pour raisonner sur des événements, des actions, des convictions, des intentions, etc. Si

on considère l'énoncé "une personne regarde la télévision", la dimension temporelle n'est pas représentée explicitement dans cette description, mais elle peut être décrite implicitement par un intervalle de temps correspondant à la durée de réalisation d'un événement.

Allen introduit une trichotomie de symboles : Propriété (*Holds*), événement (*Occurs*) et processus (*Occuring*), appliqués à un intervalle de temps durant lequel un événement se produit. Ainsi, le formalisme d'*Allen* permet :

1. D'associer une propriété p à un intervalle de temps donné i grâce au prédicat binaire *Holds*. Ce prédicat exprime le fait que la propriété p est valide si sa valeur de vérité est vraie durant tout l'intervalle de temps i. La propriété *Holds(p,i)* possède la caractéristique de l'homogénéité, ce qui signifie que si la valeur de vérité d'une propriété p est vraie sur un intervalle i, alors elle l'est sur tout sous-intervalle i' de i ;
2. De modéliser l'occurrence d'un événement e à un instant t à l'aide du prédicat *Occurs(e, t)* ;
3. De modéliser à l'aide du prédicat *Occuring (p,t)* qu'un processus p se déroule pendant une période t. Ce prédicat a été introduit pour raisonner sur des activités répétitives ;

3.7.2 Le calcul des situations "Situation calculus"

Le calcul de situations, introduit par *McCarthy* [85] et formalisé par *Levesque* [72] et *Reiter* [114], est une logique où le temps est réifié. Il est considéré comme un langage de la logique du second ordre, permettant d'axiomatiser les notions de temps, de situations et d'actions. Dans ce modèle, le monde est représenté comme un ensemble de situations ou d'états. Une situation est décrite par un ensemble de faits (fluents[18]). Elle est définie comme un intervalle de temps pendant lequel aucun changement de l'environnement ne se produit. Par conséquent, à partir d'une situation initiale *s*, l'exécution d'une action engendre une nouvelle situation *s'*. Chaque action est représentée à l'aide d'une fonction symbolique. *Open(door1, t1)*, signifie par exemple l'ouverture de la porte *door1* à un instant *t1*. Le tableau 3.5 présente des exemples d'événements ((2), (4) , (8)), de fluents ((1), (9)) et de situations ((3) ,(7), (5)).

Bien que plusieurs séquences d'actions puissent être exécutées, rien ne garantit que tous les fluents soient modifiés. Par conséquent, la mise en œuvre du "calcul situationnel" pose un problème communément connu sous le nom de "problème de Frame" (*frame problem*). Il s'agit de connaître ce qui ne change pas lorsqu'une action a lieu [83]. Ce problème est considéré comme un obstacle pour la modélisation d'un

18. Les fluents sont des expressions valuées qui décrivent des propriétés d'objets et dont l'interprétation varie dans le temps. Quand l'objet acquiert cette propriété spécifique, le fluent est dit valide. Les états des fluents sont définis par des évènements qui les initient ou les terminent.

environnement dynamique, puisqu'il s'agit de modéliser les valeurs de propriétés qui ne changent pas quand un événement/action se produit. Le problème de Frame a été en partie résolu à l'aide du formalisme de calcul des événements.

(1) John est seul dans la maison ;
(2) John a ouvert le robinet ;
(3) John prend sa douche ;
(4) Quelqu'un a sonné ;
(5) John a cessé de prendre sa douche ;
(6) John déverrouille la porte d'entrée de l'intérieur ;
(7) John reprend sa douche ;
(8) David ouvre la porte du réfrigérateur ;
(9) John est en compagnie d'une personne inconnue ;

TABLE 3.5 – Exemples d'événements, fluents et situations

3.7.3 Le calcul des événements

Le calcul des événements (event calculus), proposé par *Kowalski* [63], est une approche permettant de représenter les événements et de raisonner sur leurs effets.

Les primitives temporelles du calcul des événements sont : L'événement, le fluent, la période et les actions. La période correspond à une relation entre un intervalle de temps et une propriété du monde dont la persistance est garantit dans cet intervalle. L'occurrence d'un événement initialise ou termine la validité d'une propriété dans un intervalle de temps.

Le calcul des évènements est considéré comme une approche plus expressive que le calcul des situations, et permet de résoudre le problème de Frame grâce au mécanisme de circonscription [79].

Le calcul des événements intègre aussi le raisonnement par défaut qui peut être formalisé par la négation par l'échec. Ce raisonnement introduit, par *Reiter*, est considéré comme une approche de raisonnement non monotone [113]. Il est ainsi possible d'établir des règles d'inférences permettant d'effectuer un raisonnement par défaut, à l'image du raisonnement humain. Ce type de raisonnement permet d'aboutir à des conclusions dans le cas où l'information est incomplète.

Différents dialectes ont été proposés pour la formulation du calcul des événements [133] [90]. Le dialecte proposé par *Shanahan*, considéré comme le plus simple, comprends les prédicats suivants :

- *Initiates* (α, γ , *t*) : L'événement α rend vrai le fluent γ à partir de l'instant t ;
- *Terminates* (α, γ , *t*) : L'événement α rend faux le fluent γ à partir de l'instant t, t1 <t2 ;

- *Happens (α, t)* : L'événement α se produit à l'instant t ;
- *HoldsAt(α, t)* : Le fluent α est vrai à l'instant t ;
- *Clipped(t1 , b, t2)* : Le fluent b s'est terminé entre les deux instants t1 et t2 ;

3.7.4 Approches ontologiques pour la modélisation et la manipulation des connaissances temporelles

Nous avons recensé plusieurs travaux de l'état de l'art qui ont visé la proposition de canevas standard pour la description sémantique des connaissances temporelles. La majorité de ces travaux est basée sur la logique temporelle d'*Allen*. Parmi les travaux les plus représentatifs, nous pouvons citer le langage de marquage TimeML, les ontologies temporelles exprimées à l'aide des deux principaux langages du Web sémantique RDF et OWL, les ontologies des fluents 4D et les logiques de description temporelle.

TimeML est un langage de marquage standard de l'ISO pour représenter des évènements, des états et leurs relations temporelles dans un format structuré, à partir des énoncés textuels exprimés en langage naturel. Ce langage permet l'identification et l'ancrage des événements dans le temps grâce à l'utilisation de graphes orientés [110]. Pour ce faire, TimeML est composé de quatre balises d'annotations : Event, TIMEX3, Signal et TLINK. La balise Event est utilisée en général pour annoter des énoncés dans un texte, correspondant à des événements. La balise TIMEX3 est utilisée principalement pour marquer des expressions temporelles explicites, comme les heures, les dates, les durées, etc. La balise SIGNAL représente un signal temporel qui correspond à des mots fonctionnels qui suggèrent une relation temporelle particulière. Comme exemples de la balise SIGNAL, nous pouvant citer : Quand, dans, après. La balise TLINK permet de représenter les liens entre deux ou plusieurs événements dans le but de les ordonner dans le temps. Les balises TLINK sont les plus répandues car elles montrent comment les balises TimeML (événements et expressions temporelles TIMEX3) sont temporellement liées les unes aux autres.

La représentation des ontologies temporelles avec les deux principaux langages du Web sémantique RDF et OWL a fait l'objet de plusieurs travaux. Plusieurs extensions au langage RDF ont été proposées pour faciliter la représentation des entités temporelles dans les ontologies RDF. Ainsi, *Gutierrez et al.* proposent une extension du langage RDF permettant d'encoder l'intervalle de temps dans lequel un triplet RDF est valide [49]. Par exemple, la notation (a ; b ; c) : [t1 t2] est utilisée pour indiquer que le triplet (a ; b ; c) est valide dans l'intervalle de temps [t1 t2]. Dans [143], *Tappolet et al.* proposent une extension du langage de requêtes SPARQL, appelée T-SPARQL, pour exprimer des requêtes temporelles. Ces dernières sont converties automatiquement en requêtes SPARQL. Les requêtes T-SPARQL peuvent être soit un filtre exprimant une requête ponctuelle sur un graphe RDF valide soit à l'instant T ou durant un intervalle temporel [?s, ?e].

Le langage de requêtes SPARQL-ST, proposé par *Perry et al.* [105], permet de formuler des requêtes sur les graphes RDF spatio-temporels grâce à l'utilisation de deux types de variables, en l'occurrence, des variables temporelles identifiables par le préfixe #, et des variables spatiales identifiables avec le préfixe %. Le langage SPARQL-ST permet d'inclure des filtres temporels et spatiaux permettant de définir des conditions sur les intervalles de validité temporelle et les propriétés spatiales des triplets RDF. Une propriété spatiale correspond à un objet spatial, par exemple, un rectangle, un point, une ligne ou un polygone, décrit à l'aide de l'ontologie *GeoRSS* [19].

Les deux ontologies OWL les plus représentatives qui décrivent comment représenter et extraire des connaissances temporelles à partir des ontologies sont OWL-Time, et SWRL Temporal Ontology. Cependant, ces deux ontologies ne permettent pas de modéliser les relations entres événements. Par conséquent, elles ne répondent pas aux exigences de modélisation des changements temporels. OWL-Time (anciennement DAML-Time) est une ontologie W3C qui permet de décrire des connaissances temporelles. Elle est basée sur les concepts *Instant* et *Intervalle* et sur les relations temporelles d'Allen. Par exemple, la propriété *Inside* indique qu'un instant appartient à un intervalle donné.

SWRL Temporal Ontology, extension du langage SWRL et de la bibliothèque SWRLTab, permet l'annotation, le raisonnement et l'interrogation des bases de connaissances temporelles. Dans cette ontologie, l'annotation temporelle des connaissances est basée sur les concepts suivants : Proposition, instant et intervalle de temps qui décrivent respectivement la validité, la granularité temporelle et la durée de validité d'une proposition. La représentation des connaissances à l'aide des ontologies "SWRL Temporal Ontology" et "OWL Time" reste assez complexe. Ces deux ontologies sont plus adaptées pour la description d'entités temporelles que pour la description du contexte temporelle des connaissances ontologiques.

Parmi les approches les plus représentatives, on peut citer, l'approche de Réification, l'ontologie de Fluents 4-D [151] et la logique de description temporelle [6], [80]. Les deux premières approches peuvent être utilisées sans modification du langage OWL. La troisième approche implique, quant à elle, une extension de la logique de description, et par conséquent, la modification du langage OWL.

La réification dans le cadre de la représentation sémantique des connaissances avec des ontologies OWL consiste à représenter des relations n-aire à l'aide de prédicats u-naires ou binaires. Dans le contexte de la représentation des connaissances temporelles, la réification permet d'introduire des entités temporelles sous

19. GeoRSS est une recommandation du consortium W3C. Elle représente un vocabulaire de référence pour la description des propriétés géo spatiales des ressources Web

forme d'arguments dans les prédicats décrivant la relation d'une entité avec son intervalle ou instant de validité. Par exemple, une relation n-aire *R1* entre deux objets *A* et *B* à un moment *t*, peut être exprimée sous la forme *R1(A,B,t)*, ce qui revient à considérer la création d'une instance OWL avec *R, A, B,* et *t* comme propriétés. Prenons l'exemple de la relation *locatedAt(Person, Place, TimeInterval)* permettant d'exprimer qu'une personne se trouve à un endroit précis durant un intervalle de temps donné. La réification présente deux limitations principales : La première concerne la redondance des connaissances, puisqu'à chaque fois, une nouvelle instance est créée pour décrire une relation temporelle entre deux objets. Il s'agit d'un problème commun à toutes les approches basées sur la logique de description. La deuxième limitation concerne l'impossibilité d'associer une sémantique aux propriétés des relations, compte tenu que la structure binaire des propriétés OWL qui impose que ces propriétés soient représentées séparément de la relation elle-même.

Welty et *Fikes* ont proposé, une ontologie de Fluents 4D permettant de décrire en langage OWL les informations temporelles en utilisant des Fluents [151]. L'ontologie de fluents 4D reprise par *Batsakis* [14] considère deux types de représentations spatio-temporelles :(i) une représentation tridimensionnelle d'un objet constituée uniquement de propriétés spatiales, qui changent au cours du temps ; (ii) une représentation quadri-dimensionnelle, qui ajoute à la représentation tridimensionnelle des propriétés de l'objet une dimension temporelle grâce à l'utilisation des classes *TimeSlice* et *TimeInterval*. La représentation du changement des valeurs de vérité des propriétés d'un objet s'effectue à travers l'utilisation de fluents qui sont des propriétés valides uniquement pendant un intervalle de temps donné.

L'approche de *Batsakis* permet d'enrichir la représentation des intervalles de temps avec des prédicats décrivant les relations entre les intervalles de temps, comme par exemple, les prédicats "avant" et "après" [14]. Cette approche permet également de traduire les relations entre les intervalles de temps en relations entre leurs équivalents en instants temporels. TOWL "Time-determined Ontology Web Language" a été proposée comme alternative à OWL-Time et SWRL Temporal Ontology, pour la représentation du temps, des changements et des transitions d'état.

Bien que les ontologies Fluent 4D offrent un niveau d'expressivité élevé pour décrire les événements, les changements et les transitions dans le temps, elles présentent quelques inconvénients : i) elles engendrent une description redondante des connaissances temporelles ; ii) elles nécessitent une certaine réécriture des ontologies décrivant le domaine applicatif pour modéliser la dimension temporelle ; et iii) elles introduisent de nouvelles relations et objets pour modéliser l'information temporelle associée à chaque entité. En conséquence, l'interrogation des informations temporelles peut être lente et le raisonnement sur la sémantique des relations temporelles ne peut s'effectuer en utilisant les moteurs de raisonnement OWL-DL

classiques tels que Jena, Pellet, etc.

Pour répondre à la limitation du raisonnement temporel dans OWL-DL, et éviter d'utiliser un système de raisonnement à base de règles qui soit externe à un raisonneur de logique de description, l'idée consiste à étendre la logique de description avec des constructeurs de relations temporelles telles que "toujours dans le passé", "dans le futur", etc. Cette approche impose d'étendre la syntaxe et la sémantique d'OWL DL avec d'autres constructeurs temporels. Une variété de logiques de description temporelles ont été proposées dans la littérature, mais, à notre connaissance, il n'existe toujours pas de consensus sur une approche pouvant être prise en compte dans le standard OWL. Par ailleurs, il existe peu de travaux relatifs à l'élaboration de langages de requêtes et de règles d'inférence temporelles basés sur les logiques de description temporelles. Parmi ces travaux, nous pouvons citer "TL-OWL"[20], un langage d'ontologie temporelle basé sur la notion d'intervalle de temps [125].

3.8 Synthèse

Dans ce chapitre, nous avons dressé un panorama des formalismes de représentation symbolique des connaissances et des modèles de raisonnement associés en les situant par rapport au contexte applicatif de la thèse. Nous avons ainsi présenté et analysé les différents modèles de représentation et de raisonnement à base d'ontologies. Actuellement, RDF-S et OWL sont les langages d'ontologies les plus utilisés pour la mise en œuvre de plateformes de gestion de connaissances en intelligence ambiante et en robotique. Le langage OWL reprend la syntaxe du langage RDF-S et offre un vocabulaire plus riche pour décrire les propriétés, les classes et les relations entre classes. Il est fondé sur un fragment décidable de la logique des prédicats et comporte un ensemble de constructeurs permettant de décrire des classes complexes, comme la conjonction de classes, ou le quantificateur existentiel. Le langage standard SPARQL est généralement utilisé pour l'interrogation des bases de connaissances OWL représentées sous forme de triplets RDF.

Le raisonnement associé au langage OWL repose d'une part, sur des mécanismes d'inférence de la logique de description et d'autre part, sur des clauses de Horn exprimées en langage de règles SWRL. En pratique, ce dernier ne permet que la création par assertion de nouvelles instances de concepts ou de propriétés. L'hypothèse du monde ouvert sur laquelle sont fondés SWRL et OWL ne permet pas de supporter la négation de faits, et par conséquent, ne permet ni la modification, ni la suppression des instances durant le processus d'inférence. Dans ce cadre, nous citons les travaux de *Ian Horrocks* [136], *Riboni et al.* [167] et *Tao et al.* [163], qui ont montré, à travers des scenarii concrets, la différence des décisions prises par

20. http://cordis.europa.eu/ist/kct/towl_synopsis.htm

un système, face aux mêmes situations, selon l'hypothèse adoptée : Monde fermé ou monde ouvert. Les auteurs ont ainsi démontré que dans un contexte applicatif comme celui de l'intelligence ambiante ou de la robotique ubiquitaire, l'utilisation du langage de règles SWRL, n'est pas adaptée pour un raisonnement réactif sur des connaissances dynamiques dont la valeur de vérité peut évoluer dans le temps. Un tel raisonnement peut conduire dans certains cas à des déductions erronées ou contradictoires. Plusieurs travaux on tenté d'étendre la syntaxe et la logique de description sur lesquelles s'appuie le langage OWL [144] [175] [162]. Cependant, la prise en compte de ces extensions dans la version standard du langage OWL impose de modifier la majorité des outils existants, qui sont dans l'ensemble conçus pour des applications d'extraction de connaissances dans le web nécessitant un raisonnement selon l'hypothèse du monde ouvert.

Le contexte applicatif de nos travaux impose d'effectuer des raisonnements selon l'hypothèse du monde fermé. Par conséquent, une approche intéressante consisterait à combiner le raisonnement non-monotone avec une représentation ontologique des connaissances. L'approche hybride proposée dans le chapitre 4 vise à exploiter d'une part, les avantages des ontologies en termes d'expressivité, d'extensibilité et d'interopérabilité sémantique, et d'autre part, le raisonnement non-monotone qui supporte la négation de faits et permet de modifier ou de supprimer des connaissances, et par conséquent, de garantir une prise de décision cohérente.

Une problématique importante traitée dans cette thèse concerne la gestion des connaissances dynamiques du contexte qui nécessite de prendre en compte l'aspect temporel de ces connaissances. Dans ce cadre, nous avons recensé plusieurs approches de l'état de l'art qui ont visé la proposition d'ontologies pour la description sémantique des connaissances temporelles. Ces approches s'appuient en général sur la logique temporelle d'Allen et le paradigme des fluents. Bien que ces approches offrent un niveau d'expressivité élevé, leurs inconvénients majeurs résident dans le fait qu'elles engendrent une description redondante des connaissances temporelles et qu'elles nécessitent une réécriture de toutes les ontologies existantes. Par ailleurs, le raisonnement sur la sémantique des relations temporelles exprimées avec ces ontologies ne peut s'effectuer avec les moteurs de raisonnement OWL-DL classiques. Les inconvénients énumérés ci-dessus sont dus principalement à la difficulté de définir des prédicats d'une arité quelconque pour représenter la dimension temporelle des propriétés. Une solution à ce problème consisterait à utiliser des prédicats n-aires pour représenter sémantiquement des connaissances dynamiques et leurs liens chronologiques. Le modèle de représentation de connaissances et de raisonnement NKRL (Narrative Knowledge Representation Language) que nous présentons dans le chapitre 5 est particulièrement bien adapté pour la reconnaissance de contextes non-triviaux pouvant être liés au temps [119] [120]. Ce modèle offre un haut niveau d'expressivité et permet la modélisation de connaissances narratives à partir des ontologies HClass et HTemp décrivant respectivement une hiérarchie de concepts et une hiérarchie d'événements.

Chapitre 4

Modélisation sémantique et raisonnement réactif

Sommaire

4.1 Introduction

Dans ce chapitre, nous présentons le modèle sémantique proposé pour la représentation des connaissances contextuelles et le raisonnement réactif. Ce modèle, conçu pour répondre aux contraintes imposées par les applications d'intelligence ambiante, s'appuie sur deux langages : Le langage d'ontologie μConcept et le langage de règles *SmartRules*. Dans la première partie du chapitre, nous présentons les formalismes de ces deux langages.

Dans la suite du chapitre, nous présentons l'ontologie *AmiOnt*, que nous avons proposée et développée à partir du langage μConcept. Cette ontologie permet de décrire tous les concepts nécessaires et tous les objets qui peuplent un environnement d'intelligence ambiante (humains, robots, capteurs, actionneurs, robots, etc) et toute information, action ou événement pouvant être observée à partir de capteurs logiques ou physiques.

Dans la dernière partie du chapitre, nous montrons comment la reconnaissance implicite de contextes et l'adaptation au contexte s'effectuent à partir d'un ensemble de règles *SmartRules* exprimées à partir de l'ontologie *AmiOnt*.

4.2 Formalisme du langage μConcept

Le langage μConcept, basé sur le principe du monde fermé avec supposition du nom unique, offre un moyen de représenter sémantiquement des objets physiques ou logiques présents dans l'environnement (robots, capteurs, applications, etc.). Pour certaines classes d'objets, il permet aussi de représenter des actions exécutées par ces objets.

Définition 1 : Un μConcept est une représentation sémantique formelle d'une entité physique ou logique, telle qu'une personne, un robot, un capteur, un actionneur, etc. Il permet de définir les caractéristiques d'une entité, de définir des groupes d'entités partageant des caractéristiques communes, de déclarer les actions pouvant être exécutées par ces entités, et offre un support de raisonnement sur ces entités. La description d'une entité dans le formalisme μConcept est exprimée via des propriétés et des relations. Ces dernières sont définies indépendamment des concepts manipulés. Elles symbolisent des liens pour décrire les relations d'une part, entre concepts et d'autre part, entre un concept et une valeur littérale, par exemple, un entier ou une chaîne de caractères. Pour ce faire, le langage μConcept offre un mécanisme permettant d'annoter les concepts, propriétés, actions et instances. Une des caractéristiques importantes offerte par le langage μConcept est la possibilité de définir des actions/commandes que chaque instance d'une entité du modèle sémantique est en mesure d'exécuter.

Exemple. Le concept *Stove* est exprimé en langage μConcept comme suit :
<smc :Concept rdf :ID = "Stove">
</smc :Concept>

Dans les règles de nommage du langage μConcept, l'identifiant d'un concept est une chaîne de caractères qui commence par une majuscule. Le caractère de soulignement est utilisé pour nommer un concept composé de plusieurs termes, comme par exemple, Space_Region, Home_Control_System. Les instances μConcept sont nommées en majuscule, comme par exemple, ROBOT_KOMPAI. Enfin, l'identifiant d'une action est une chaîne de caractères qui commence par le caractère de soulignement, comme par exemple, _SwitchOff, _Open.

4.2.1 Lien entre le formalisme μConcept et les autres standards

Dans le domaine du web sémantique, Resource Description Framework (RDF) et RDF-Schema (RDF-S) sont considérés comme des langages standards compte tenu de leurs propriétés d'extension et d'interopérabilité. Nous avons fait le choix de construire le langage μConcept au dessus de ces normes, figure 4.1. En conséquence, ce modèle est sémantiquement et syntaxiquement compatible avec la norme RDF/RDF-S. La seule exception étant que les instances et les classes sont disjointes. En plus de la description du langage μConcept, une représentation du langage μConcept en RDF-S a été définie. Elle permet l'utilisation du formalisme μConcept

avec les outils RDF/RDF-S du marché, et ainsi, de réutiliser des ontologies existantes.

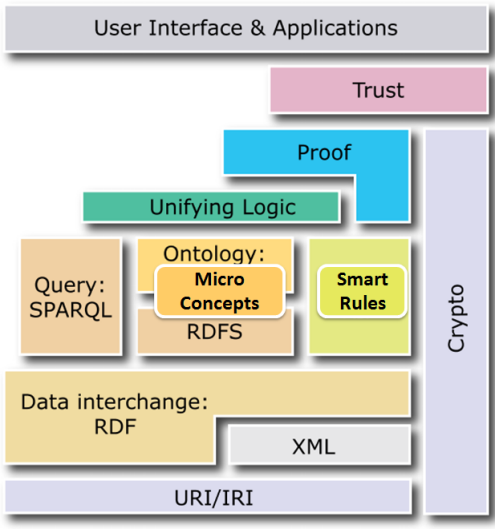

FIGURE 4.1 – Relations entre le langage μConcept et les standards existants

4.2.2 Représentation formelle du langage μConcept

Dans ce qui suit, nous décrivons le modèle μConcept à travers l'ensemble des définitions suivantes :

Définition 2. Le langage μConcept est construit autour des notions de base suivantes :

- Concept : Un concept correspond ici à la notion de classe dans le web sémantique ;
- Relation (ou Propriété) : La notion de propriété est ici similaire à celle manipulée dans le web sémantique ;
- Instance : La notion d'instance correspond à la notion d'individu dans le web sémantique ;
- Action : Il n'existe pas de correspondance dans le standard web sémantique ;

Ainsi, une base de connaissances utilisant le langage μConcept peut être représentée sous forme d'un n-uplet :

$$\langle C, P, In^P, A^{In} \rangle \tag{4.1}$$

Où

- C définit un ensemble d'entités partageant des caractéristiques communes. Chaque entité est décrite par un ensemble de propriétés P ;
- A^I représente les actions qu'une instance In peut exécuter ;
- Les propriétés sont indépendantes des concepts, formellement, $P \sqcap C = \{\}$. où le symbole $\sqcap$ représente l'opérateur de conjonction.

Les propriétés ont pour rôle de décrire les concepts et de définir respectivement des relations (R) entre :
- Deux concepts : *R(C,C)*
- Un concept et ses instances : *R(C,In)*
- Une instance et une valeur littérale : *R(In, v)*

– In^P : Une instance *In* associée à un concept. Elle peut avoir plusieurs propriétés, P =*p1,p2,p3,...,pn* /pi ∈P | $i \in [1, n]$.

Exemples de Propriétés :

1. *switchOn* : Propriété booléenne décrivant l'état fonctionnel d'une entité telle qu'une cuisinière ;
2. *isMoving* : Propriété booléenne précisant qu'une entité telle qu'une *personne*, *robot* ou *animal* est en mouvement ;
3. *hasLocation* : Propriété permettant d'exprimer qu'une entité (personne, lecteur RFID, caméra, chaise, etc.) se trouve dans un espace donné ;
4. *hasPosture* : Propriété décrivant la posture d'une entité. Cette propriété peut avoir pour valeur un label : Allongée, debout, assise, etc ;

Définition 3. Soient TC, TA et TR respectivement l'ensemble des noms de concepts, d'actions et de relations (propriétés) appartenant à un domaine donné. L'interprétation $I = (\triangle^I, .^I)$ définit une sémantique formelle des termes TC, TA et TP où $\triangle^I$ (ensemble d'instances) est le domaine de l'interprétation de I.
$.^I$ est la fonction d'interprétation qui fait correspondre d'une part, chaque nom de concept a $\in TC$ à un sous-ensemble d'éléments de $\triangle^I$ tel que : a $\in TC$ si a $I \sqsubseteq \triangle^I$ et d'autre part, chaque rôle TR à un sous-ensemble de $\triangle^I$ x $\triangle^I$.

Définition 4. Les concepts sont ordonnés par la relation notée $\sqsubseteq$. Pour tout concept E et F $\in TC$, on dit que F est plus général que E. On dit aussi que, F subsume E, ou que E est plus spécifique que F, et on écrit $E \sqsubseteq F$. Une interprétation I est un modèle $E \sqsubseteq F$ tel que $E^I \sqsubseteq F^I$.

Définition 5. La relation de subsomption dans le langage μConcept est non-réflexive, antisymétrique, transitive et acyclique.
Soient les concepts E, F, $D \in TC$, par conséquent, si $E \sqsubseteq F$, $F \sqsubseteq D$ alors $D \not\subseteq$ E.

Exemple :
À l'aide du modèle sémantique proposé, la représentation du concept *Indoor_Space_Region* comme sous-concept du concept *Space_Region* s'exprime comme suit :
<smc :Concept rdf :ID = "Indoor_Space_Region">
<rdfs :subClassOf rdf :resource = "Space_Region"/>
</smc :Concept>

Définition 6. La représentation $(a : E)$ décrit une instance (un fait) du concept E ($E \in$ TC). Ainsi, une interprétation I est un modèle d'assertion $a : E$ si $a^I \in E^I$.

Définition 7. Les relations dans le langage proposé sont de type binaire. Soient deux concepts E et F $\in TC$, a une instance de E et b une instance de F.
Une relation r $\in TR$ telle que r $\subseteq \triangle^I$ x $\triangle^I$ est définie comme une relation :
- soit sur a et b, et s'écrit r(a,b).
- soit sur une valeur v et alors elle s'écrit r.{v }. Formellement,
une interprétation I est un modèle d'assertion r (a,b) si $(a^I , b^I) \in r^I$ et $v \in \triangle^I$.

Définition 8. Le modèle proposé permet de déclarer des restrictions de cardinalité min ($\geqslant$ nR) ou max ($\geqslant$ nR) sur un concept/sous-concept et de définir une relation (propriété) fonctionnelle entre concepts R($\top \sqsubseteq (\geqslant 1$ R)), où (R $\in TR$) et $\top$ désigne un concept universel. Sémantiquement, cela signifie que :
$\{a \mid \exists b.(a, b) \in R^I \} \geqslant$ n et $\{ a \mid \exists b.(a, b) \in R^I \} \leqslant$ m.

Dans le modèle proposé, les propriétés ont des noms uniques. Si les propriétés ont le même nom dans deux concepts hérités, elles sont fusionnées dans la même propriété. Si des actions ont le même nom dans deux concepts hérités, elles sont considérées comme différentes. Ainsi, si deux concepts parents définissent des restrictions sur des propriétés alors ces restrictions sont fusionnées dans un même sous-concept.

Définition 9. À l'image d'OWL 2, le langage μConcept permet de définir la propriété de restriction sur les propriétés, "*Qualified number restriction*". Formellement : ($\geqslant$ nR.C , $\leqslant$ nR.C) où C $\in TC$ est un concept et R $\in TR$ une relation :
$\{a \mid \exists b.(a, b) \in R^I$ et b $\in C^I \} \geqslant$ n et $\{ a \mid \exists b.(a, b) \in R^I$ et b $\in C^I \} \leqslant$ n.

Définition 10. La description de l'état d'une instance dans la base de connaissances est décrite par les valeurs de ses propriétés. Par exemple, à l'aide du symbole d'égalité (==) et du symbole d'affection (:=), les informations indiquant que la maison est vide, que la personne n'est pas à l'intérieur de son habitation et que la personne porte un tag RFID, sont décrites respectivement comme suit :

House (isOccupied == true)
Person (isOutSide == true)
Person (hasTag := Tag_RFID)

Définition 11. Les types de données du langage μConcept sont représentés en format XML-schéma. Ces données sont de simples valeurs qui ne sont pas associées à la sémantique du μConcept mais qui héritent plutôt de la sémantique associée aux spécifications du modèle XML. Ces données peuvent être utilisées dans les paramètres des actions ou comme valeurs de propriétés. Les différents types supportés par un concept sont : xsd :integer, xsd : float, xsd : boolean, xsd :int, xsd :string, xsd :dateTime, xsd :time, xsd :date, xsd :gYearMonth, xsd :gYear, xsd :gMonthDay, xsd :gDay, xsd :gMonth , xsd :duration, etc.

Définition 12. Contrairement à OWL 2, le formalisme μConcept est basé sur le principe de la présupposition du nom unique. Formellement, si a et b sont deux instances alors si a $\neq$ b $\Longrightarrow a^I \neq b^I$.

4.2.2.1 Définition des actions

Définition 13. Une des notions fondamentales sur laquelle s'appuie le langage μConcept est la définition des actions. Une action A est un quadruplet $\langle D, act, E, S\rangle$, où D $\in$ TC représente le domaine d'une instance, $act \in$ TA représente une action à exécuter, E et S sont les paramètres d'entrée/sortie de l'action. Le paramètre input est obligatoire, tandis que le paramètre output est optionnel. Formellement, on écrit : $act_{e,s}(D^I)$ | (e $\in$ E et s $\in$ S).

Notons que les propriétés de restrictions sur les valeurs des propriétés sont appliquées avant et après l'exécution d'une action. Formellement, soit a(e,s) $\in TA$, on vérifie :
{ $\forall$ (e $\in$ E et s $\in$ S) | $e^I \sqsubseteq \triangle^I$ et $s^I \sqsubseteq \triangle^I$ } et { a | $\exists$ b.(a,b) $\in R^I$ } $\geqslant$ n et { a | $\exists$ b.(a,b) $\in R.^I$ } $\leqslant$ n

Si plusieurs domaines $D_1 \cup D_2 \cup ... \cup D_n$ sont utilisés, alors chaque instance i de l'un des domaines (i $\in D_n^I$ | $D_n^I \in TC^I$) est en mesure d'effectuer l'action.

Exemples :

1. _*SwitchOn* et _*Open* : Actions pouvant être exécutées par les instances des concepts *Light_Spot_Actuator* et *Door_Lock* par exemple ;

2. _*Move* : Action pouvant être exécutée par une instance du concept *Robot* ;

3. _*Open* et _*Close* : Actions pouvant être supportées par les instances des concepts *Door_Lock* et *Shutter_Lock* par exemple ;

Exemple de déclaration d'une action avec le langage μConcept

À l'aide du formalisme μConcept, nous exprimons qu'une instance du concept *Robot* peut être déplacée (action "_*MoveRobot*") vers un "*espace*" donnée, comme suit :

```
(1) <smc :Action rdf :about="http ://www.sembysem.org/AmiOnt#_MoveRobot">
(2)   <smc :actionType rdf :datatype="http ://www.w3.org/2001/XMLSchema#string" >
(3)    Physical
(4)   </smc :actionType>
(5)  <smc :actionDomain rdf :resource="http ://www.sembysem.org/AmiOnt#Robot"/>
(6) </smc :Action>

(7)<smc :FunctionalParameter rdf :about="http ://www.sembysem.org/AmiOnt#toLocation">
(8)   <rdfs :range>
(9)      <smc :Concept rdf :about="http ://www.sembysem.org/AmiOnt#Room">
(10)        <rdfs :subClassOf>
(11)          <smc :Concept rdf :about="http ://www.sembysem.org/AmiOnt#House"/>
(12)        </rdfs :subClassOf>
(13)     </smc :Concept>
(14)  </rdfs :range>
(15)   <rdfs :domain rdf :resource="http ://www.sembysem.org/AmiOnt#_MoveRobot/Input"/>
(16) </smc :FunctionalParameter>
```

Dans l'exemple ci-dessus, les lignes de (1) à (6) permettent de définir d'une part, l'action *_MoveRobot* et d'autre part, d'exprimer que cette action ne peut être exécutée que par l'entité *Robot*. Les lignes de (7) à (16) donnent plus de précisions sur l'action à exécuter par le robot. Il s'agit ici d'indiquer l'espace où le robot doit se déplacer. Ainsi, la propriété *toLocation* correspondant au paramètre input de l'action *_MoveRobot*, indique que le robot doit se déplacer vers une chambre représentée par le concept *Room*.

Par rapport à OWL, le langage μConcept permet d'introduire, en plus du support des actions, de nombreuses contraintes et opérateurs permettant de construire des modèles plus expressifs : Opérateur d'agrégation de propriétés, valeurs par défaut, listes ordonnées.

4.2.2.2 Liste ordonnée

Le langage μConcept offre un moyen de définir une propriété non-fonctionnelle dite propriété multivaluée. Cette propriété permet de définir une liste ordonnée de propriété qu'une instance peut avoir. La déclaration d'une liste ordonnée de propriétés est définie comme suit :

```
<smc :ListProperty rdf :about="http ://www.sembysem.org/AmiOnt#allLocation">
    <rdfs :range rdf :resource="http ://www.sembysem.org/AmiOnt#Room"/>
    <rdfs :domain rdf :resource="http ://www.sembysem.org/AmiOnt#Person"/>
</smc :ListProperty>
```

4.2.2.3 Composition de propriétés avancées

Le langage μConcept possède des opérateurs de calcul (opérateurs arithmétiques, comptage, etc.) sur les valeurs de propriétés : *sum, minimum, maximum, count et average.*

À titre d'exemple, l'instruction ci-dessous permet de calculer la température de la maison à partir de l'ensemble des températures relevées dans une pièce,

- Habitat.temperature = count (Room.temperature)

4.2.2.4 Valeur par défaut

En langage μConcept, il est possible d'assigner une valeur par défaut à chaque propriété d'une instance lors de sa création. Un sous-concept peut redéfinir la valeur par défaut d'une propriété héritée et dans ce cas, la valeur du concept parent est ignorée.

Exemple de déclaration de valeur par défaut en langage μConcept :

```
<smc :Concept rdf :ID = "Stove">
    <smc :restriction>
    <smc :PropertyRestriction>
    <smc :onProperty rdf :resource = "switchOn"/>
    <smc :default rdf :datatype="xsd ;boolean">false</smc :default>
    </smc :PropertyRestriction>
    </smc :restriction>
</smc :Concept>
```

4.2.2.5 Logique d'inférence

Une des principales caractéristiques du mécanisme d'inférence associé au langage μConcept qui adopte l'hypothèse du monde fermé (CWA), consiste à vérifier que les contraintes définies dans le modèle sémantique sont satisfaites avant la mise à jour de la base de connaissances. Ainsi, si un fait est inconnu car inexistant dans la base de connaissances, il est considéré comme faux.

Manipuler des connaissances complètes sur l'environnement permet un contrôle plus sûr du système. Le contrôle est assuré, à travers un ensemble de règles combinant des concepts et des propriétés. Ces règles manipulent uniquement les faits insérés dans la base de connaissances. Elles permettent d'exprimer un raisonnement de haut niveau où le système raisonne sur la représentation sémantique des entités de l'environnement. Cette approche à base de règles permet de garantir que le modèle sémantique représente le dernier état d'une entité physique ou logique de l'environnement. Elle permet aussi de simplifier l'écriture des règles de raisonnement et d'augmenter le niveau d'abstraction dans la gestion et le contrôle de l'environnement. En effet, dans le langage μConcept et contrairement à OWL, lorsqu'une nouvelle valeur est assignée à la propriété d'une entité, la valeur précédente de cette propriété est écrasée. Dans ce cadre, les règles sont étroitement liées à l'ontologie,

du fait qu'elles n'utilisent que des prédicats liés aux concepts et aux propriétés (relations) de l'application.

4.3 Le langage de règles SmartRules

Dans notre approche, la gestion du contexte s'effectue en combinant des concepts et leurs relations dans un ensemble de règles sémantiques définies dans un langage de règles, appelé *SmartRules* [1] [118]. Ces règles sont exprimées à partir de l'ontologie définie pour l'application considérée.

4.3.1 Caractéristiques recherchées pour les règles

Les exigences spécifiées pour la définition de ce langage de règles sont les suivantes :

- Exprimer des règles en utilisant le vocabulaire d'une ontologie décrite avec le langage μConcept, figure 4.2 ;
- Manipuler une base de connaissances qui change continuellement ;
- Exprimer les règles de manière simple ;
- Avoir la possibilité de modifier les valeurs des instances de concepts vérifiant l'ensemble des contraintes définies dans le modèle sémantique ;
- Lancer des actions sur les instances de concept ;

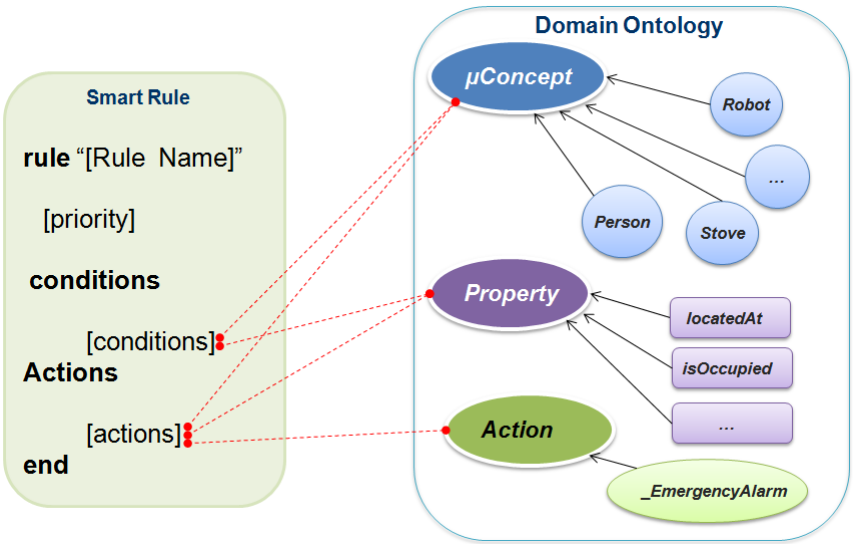

FIGURE 4.2 – Schéma général d'une règle SmartRules.

1. Les règles considérées sont de type règles de production. https://en.wikipedia.org/wiki/Production_rule

4.3.2 Formalisme du langage SmartRules

Pour répondre à l'ensemble des exigences énumérées ci-dessus et garantir la cohérence du système, le moteur de raisonnement doit être capable de prendre en compte la dynamicité du contexte. La décidabilité et la capacité à traiter une grande base de connaissances font de l'algorithme *Rete* [40] et en particulier sa version *Rete Oriente Objet* (*Rete OO*), un moteur de raisonnement particulièrement bien adapté au modèle sémantique proposé dans ce chapitre. Le langage de règles que nous proposons est indissociable du langage μConcept. Ce langage s'appuie sur le modèle de règles classique suivant :
IF condition(s) THEN operation(s)/actions(s) sur les instances.

Exemple : Supposons que dès le déclenchement d'un incendie, on désire identifier et suivre les déplacements des occupants d'un immeuble. Pour ce faire, chaque personne est équipée d'un tag $RFID$. Les lecteurs $RFID$, installés dans les parties communes et dans chaque habitation (escalier, ascenseur, couloir, toilettes, salle de bain, porte d'entrée, etc.), servent à notifier l'entrée d'une personne dans un espace donné. Si nous définissons une propriété *"allLocation"*, à chaque passage d'une personne dans la zone de couverture d'un lecteur $RFID$, sa position est enregistrée dans la base de connaissances sous la forme d'une instance d'un concept. Il est ainsi possible de tracer son déplacement en utilisant la règle "*Human Movement Tracking*" définie dans le tableau 4.1.

```
rule "Human Movement Tracking"
conditions
 ?p := Person( ?currentLocation := one (allLocation) ) ;
actions
  définir une ou plusieurs actions d'adaptation au contexte
        ....
end
```

TABLE 4.1 – Suivi d'une personne dans un habitat donné.

Dans cet exemple, le concept *Person* est le domaine de la propriété *allLocation* qui est une propriété multivaluée. Cette propriété peut avoir plusieurs valeurs pour une instance donnée. À l'aide de l'opérateur *one* utilisé uniquement avec une propriété multivaluée, la règle "*Human Movement Tracking*" pourra traiter toutes les valeurs de cette propriété. Ainsi, cette règle permet de vérifier l'existence d'abord d'une instance de *Person*, puis de déduire toutes les valeurs de cette propriété pour chacune des personnes.

4.3.2.1 Définition des règles

La syntaxe formelle d'une règle *SmartRules* est définie dans 4.2 où le symbole CP représente une ou plusieurs conditions définies dans une règle. Le symbole AP,

quant à lui, représente une ou plusieurs actions (mise à jour de propriété, action définie dans l'ontologie d'une application, etc.). Formellement :

$$\bigwedge_{i=0}^{m} CP \longrightarrow \bigwedge_{j=0}^{n} AP \tag{4.2}$$

où CP est une conjonction de conditions ($cp_0 \wedge cp_1 \wedge$ ... $cp_{m-1} \wedge cp_m$), et AP une conjonction d'actions ($ap_0 \wedge ap_1 \wedge$ ... $ap_{n-1} \wedge ap_n$).

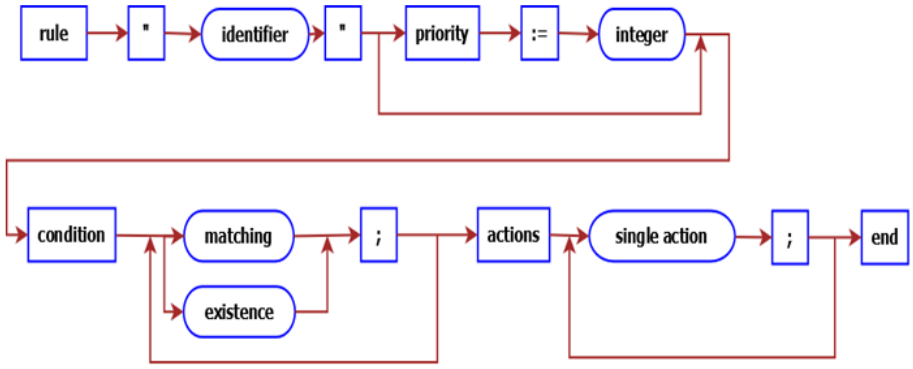

FIGURE 4.3 – Diagramme de définition d'une règle SmartRules.

```
rule "UnoccupiedHouse"

  conditions
      Living_Room (not (isOccupied)); .....c1
      Room (not (isOccupied)); ..............c2
      Kitchen(not (isOccupied)); ............c3
      Hall (not  (isOccupied)); ..............c4
      Bath_Room (not  (isOccupied)); .....c5
      ?x:= Person();  ..........................c6
 actions
      //update property value  (isAtHome)
      ?x->isAtHome :=false;      (a1)
end
```

Living_Room, Room, Hall, Kitchen et Bath_Room sont des concepts

(c6) : Chercher toutes les instances du concept Person

isAtHome: Une propriété fonctionnelle

FIGURE 4.4 – Exemple de règle SmartRules.

Le diagramme de définition d'une règle SmartRules est illustré figure 4.3. La figure 4.4 décrit un exemple de règle permettant de mettre à jour la valeur de la propriété *isAtHome* à false. Cette opération s'effectue grâce à l'action (a1) après vérification de la véracité des conditions *c1, c2, ..., c6.*

4.3.2.2 L'appariement

L'appariement consiste à chercher tous les unificateurs tels que la représentation sémantique des concepts définis dans les prémisses des règles pouvant être appariées

(unifiées) avec une ou plusieurs instances (faits) de la base de connaissances. Une prémisse peut avoir ou non des contraintes 4.3. Une contrainte Ct, définie pour un concept C, peut être simple ou composée. Notons que seules les instances d'un concept C mises à jour dans la base de connaissances sont considérées.

$$C(\bigwedge_{i=0}^{n} Ct) \tag{4.3}$$

où C et Ct appartiennent au domaine de l'ontologie.

Une prémisse sans contraintes signifie que toutes les instances du même concept sont évaluées. Ainsi dans la figure 4.4, la condition (c6) impose que toutes les instances du concept *Person* soient évaluées, tandis que les conditions (c1), ..., (c5) définissent chacune une contrainte *(isOccupied)* booléenne .

4.3.2.3 Déclaration de variable

La notion de variables dans le langage de règles *SmartRules* est similaire à celle utilisée dans les autres langages de règles, tels que *Drools*, *Jess*, etc. Une variable peut être utilisée pour stocker une connaissance (instance, valeur de propriété ou d'un littérale), figure 4.5. L'identifiant de chaque variable est défini à l'aide de l'expression suivante : $[?][a-z][a-zA-Z0-9]*$. Formellement :

$$v \longleftarrow C(\bigwedge_{i=0}^{n} Ct) \tag{4.4}$$

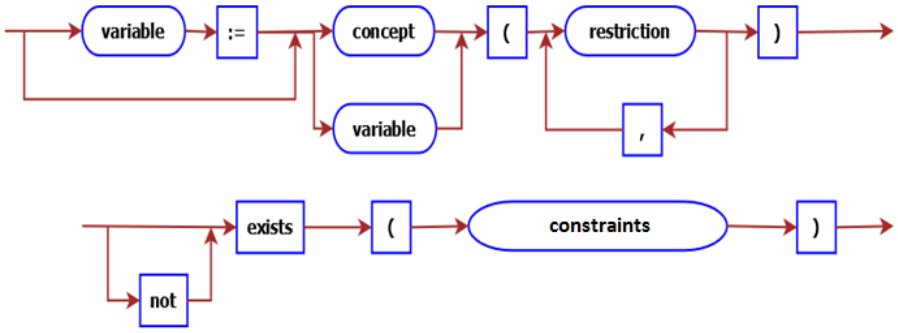

FIGURE 4.5 – Diagramme de définition de variable.

Contrairement aux langages imposant la déclaration du type de la variable, le langage *SmartRules* déduit automatiquement le type de la variable de l'élément référencé. Les variables déclarées dans une règle ne sont visibles qu'à l'intérieur de cette règle. Cependant, il n'est pas possible de déclarer une variable de même nom déjà déclarée comme variable globale. Dans la condition (c6), figure 4.4, la variable $?x$ permet de référencer toutes les instances de type *Person*.

```
rule "SafetyRule"

  conditions
      Person (not (isAtHome));          (c1)
      Stove  (isRunning, ?ID:=hasID);   (c2)

  actions
      Action  ?switchOffStove  :=  createInstance (_SwitchOff); (a1)
      ?switchOffStove  -> instanceID := ?ID;
      execute(?switchOffStove);          (a2)
end
```

FIGURE 4.7 – Exemple de règle d'adaptation au contexte

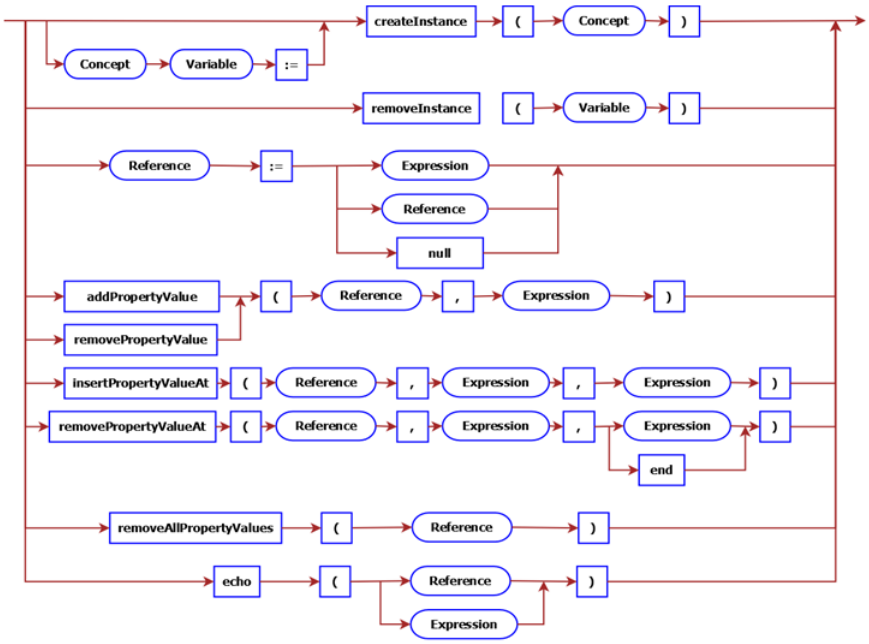

FIGURE 4.6 – Diagramme de déclaration d'action

4.3.3 Définition des actions

La partie "actions" d'une règle représente les actions à exécuter dans l'environnement réel, ou les nouvelles connaissances contextuelles inférées. Dans la version actuelle du langage *SmartRules*, il s'agit de créer, supprimer des instances ou une valeur de propriété dans la base de connaissances et/ou modifier/ajouter une valeur de propriété, figure 4.6. Par exemple, l'action *(a1)* définie dans la règle illustrée, figure 4.7, permet d'éteindre une cuisinière. Il s'agit ici d'inférer que

l'habitat est vide (c1) et que la personne a oublié d'éteindre la cuisinière *(Stove)* (c2). Le paramètre *instanceId* de l'action *_SwitchOff* indique l'instance exacte correspondant à la cuisinière à éteindre. L'adaptation à ce contexte consiste alors à éteindre cette cuisinière *(Stove)*.

4.4 Démarche de modélisation d'un environnement à intelligence ambiante avec le langage μConcept

Dans ce paragraphe, nous proposons une ontologie appelée *AmiOnt* composée de concepts suffisants pour décrire de façon générique et expressive toutes les entités qui peuplent un environnement intelligent ambiant, et toute information, action ou événement pouvant être observée à partir de capteurs logiques ou physiques. Cette ontologie vise à être un support pour établir des règles définies au moyen du langage de règles *SmartRules* pour la gestion de la sensibilité au contexte qu'il s'agisse de règles pour la reconnaissance du contexte ou de règles pour l'adaptation au contexte. L'approche de modélisation que nous proposons est souple et extensible puisqu'elle permet de mettre à jour ou d'étendre les modèles de contexte définis préalablement, par simple modification, suppression ou ajout de règles et ce, sans remettre en question les axiomes décrits dans l'ontologie *AmiOnt*.

Une propriété importante que doit posséder l'ontologie *AmiOnt* est son interopérabilité avec les ontologies universelles existantes telles que *DOLCE Ultra Lite, NKRL, OpenCyc, SUMO*, etc., figure 4.8. Néanmoins, nous nous sommes focalisés dans le cadre de cette thèse sur la réutilisation de certains concepts définis dans les ontologies *SSN, DOLCE Ultra Lite et NKRL*. L'utilisation des concepts de l'ontologie HClass-NKRL permet d'assurer d'interopérabilité sémantique des connaissances utilisée dans le raisonnement réactif avec celle utilisée dans le raisonnement narratif. L'utilisation des concepts de l'ontologie SSN permet de décrire les perceptions des capteurs. L'onologie SSN est alignée nativement sur l'ontologie de haut niveau *DOLCE Ultra Lite*. Cette dernière fournie un ensemble de concepts de haut niveau permettant de garantir l'interopérabilité entre plusieurs ontologies applicatives.

La structure globale de l'ontologie *AmIOnt* est composée de cinq concepts de haut niveau inspirés de l'ontologie *DOLCE Ultra Lite* :

1. *Object* : Il s'agit d'un concept abstrait permettant de décrire tous les objets physiques ou immatériels présents dans l'environnement. Nous distinguons trois catégories d'objets, qui sont définies respectivement par les sous-concepts suivants : *Active_ Object, Static_ Object* et *Movable_ Object*. Le concept *Active_ Object* permet de décrire d'une part, des objets physiques de type capteurs (*Sensor*), actionneurs (*Actuator*) ou dispositifs informatiques (*Device*), et d'autre part, des objets immatériels tels qu'un web service, une base de données ou un fichier. Les objets immatériels peuvent être utilisés pour mettre en œuvre des capteurs virtuels permettant de fournir des informations contex-

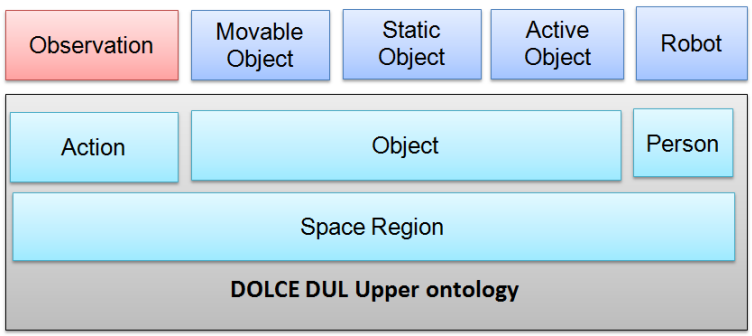

FIGURE 4.8 – Concepts de base de l'ontologie *AmiOnt* pour l'intelligence Ambiante.

tuelles, comme par exemple, les informations météo. Les concepts objets *Static_ Object* (objets statiques) ou *Movable_ Object* (objets mobiles) sont utilisés pour décrire les observations nécessaires pour les règles de gestion du contexte, et plus généralement dans la boucle perception-décision-action des applications.

2. *Person* : Il s'agit d'un concept central de l'ontologie qui permet de décrire l'identité et le profil d'un usager. Ce concept peut être spécialisé pour décrire des profils d'usagers par type d'application (profil pour une application de réseautage social, profil pour une application d'assistance cognitive, etc.) ;

3. *Robot* : Ce concept permet de décrire les caractéristiques génériques d'un robot de service, comme par exemple, l'architecture du robot en termes de capteurs et d'actionneurs. Ces derniers sont généralement associés à d'autres concepts dérivés du concept "*Active_ Object*" de l'ontologie *AmiOnt*. Le concept *Robot* peut être spécialisé pour définir des sous-classes de robots, comme par exemple un robot compagnon mobile ou un robot exosquelette, etc.

4. *Action* : Ce concept permet de décrire toutes les actions d'adaptation au contexte. La relation entre le concept *Action* et le concept *Actuator* permet d'associer une action à un actionneur, comme par exemple, diffuser un message vocal en utilisant un robot comme média de communication, ou afficher ce message sur l'écran d'un *Smartphone*.

5. *Space_ Region* : Le concept *Space_ Region* permet de décrire l'espace de localisation de personnes, objets, robots, actions, interactions, événements, etc. Le concept *Space_ Region* qui hérite du concept générique *Region*, il est différent des concepts *DUL Place* et *DUL SpatioTemporalRegion*. Ces derniers sont utilisés pour décrire des lieux ayant une signification sociale ou spatio-temporelle, comme par exemple un lieu de travail ou de réunion, etc.

4.4.1 Description spatiale d'un environnement ambiant

Il s'agit ici de réutiliser les concepts de sens commun définis dans l'ontologie *DOLCE Ultra Lite* pour la représentation spatiale de l'environnement ambiant, figure 4.9. Ainsi, à partir du concept de haut niveau *Building_ Area_ Component* héritant du concept *Space_ Region*, nous définissons une taxonomie de concepts permettant de modéliser un habitat et ses différents espaces, figure 4.9. Cette taxonomie comprend un concept racine (*House*) et les sous-concepts suivants : *Indoor_ Space_ Region* pour décrire des espaces internes de la l'habitat et *Outdoor_ Space_ Region* pour décrire des espaces externes. De ces deux concepts héritent des concepts d'espaces internes ou externes plus spécifiques, comme par exemple : *Room, Bath_ Room, Living_ Room, Garden,* etc., figure 4.10.

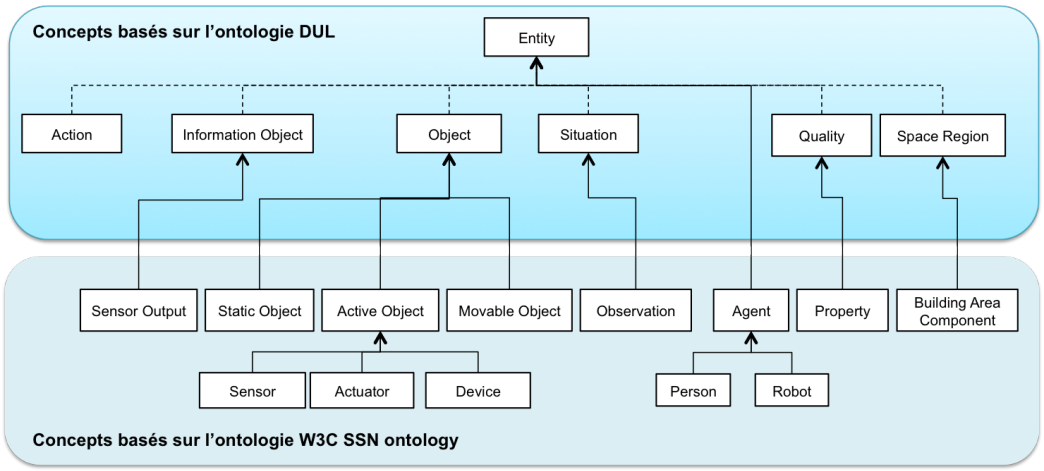

FIGURE 4.9 – Alignement des concepts de l'ontologie *AmiOnt* avec ceux de l'ontologie *DOLCE Ultra Lite*.

À ces concepts, nous avons associé deux types de propriétés pour représenter des relations spatiales d'une part, entre des objets et des espaces, et d'autre part, entre des espaces. Sans être exhaustif, nous pouvons citer les propriétés suivantes :

-*hasLocation* : Il s'agit d'une propriété générique qui permet de décrire la position relative ou absolue d'une entité. La propriété *hasRegion* peut être utilisée de manière similaire pour décrire la relation entre une entité et une region.

-*localtedAt* : Il s'agit d'une propriété similaire à *hasLocation* mais son usage est restreint à la description de la position d'un robot ou d'une personne. Exemple : Le robot se trouve dans le séjour ;

-*nearTo et farFrom* : Ces deux propriétés permettent de décrire la distance entre deux entité génériques. Par exemple, le Robot est proche de la cuisine, John est proche du téléviseur, le robot est proche de l'espace où se trouve John, etc ;

-*isAccessibleSpace* : Cette propriété décrit l'état d'accessibilité d'un espace pour un objet mobile. Par exemple, la porte donnant accès à la chambre est ouverte.

-*precedes* : Il s'agit d'une propriété qui permet de décrire une relation de recou-

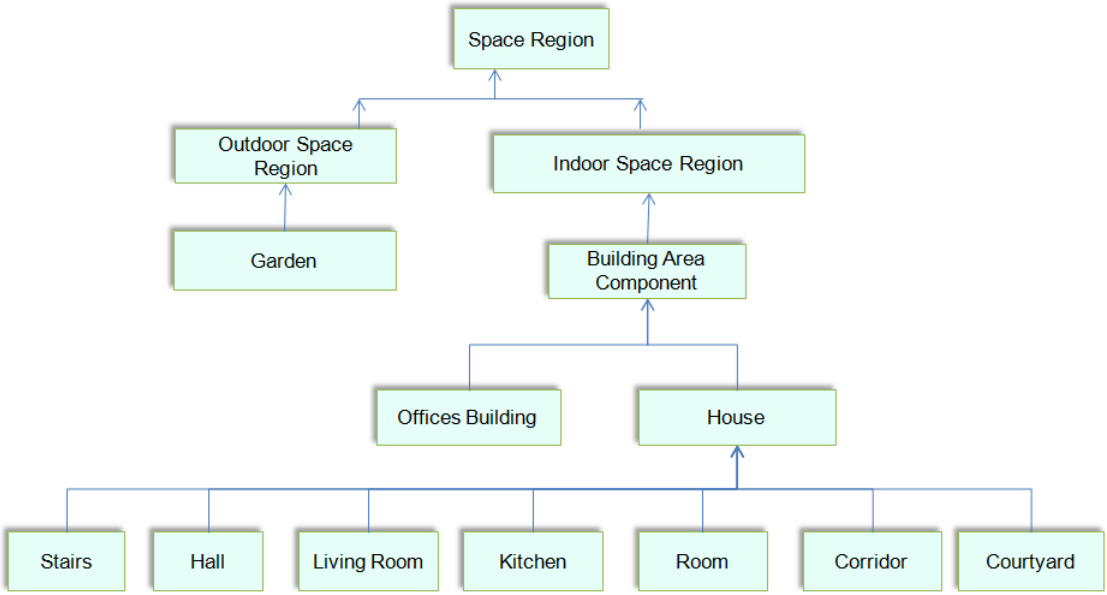

FIGURE 4.10 – Vue partielle de la description spatiale d'un environnement ambiant.

vrement ou de chevauchement entre des regions ou des intervalles de temps, comme par exemple : L'espace cuisine recouvre l'espace du hall d'entrée ;

4.4.2 Description des objets capteurs et actionneurs

Dans ce paragraphe, nous nous focalisons sur la description des capteurs et des actionneurs. Pour ce faire, nous avons réutilisé les concepts et propriétés de l'ontologie *SSN* alignée sur l'ontologie de haut niveau *DOLCE Ultra Lite*. Par exemple, le concept *Sensor*, qui hérite du concept *ActiveObject*, possède deux propriétés de base *observes* et *produces*. La propriété *observes* permet de décrire les propriétés qualitatives d'un capteur comme par exemple : Sa précision, sa résolution, son temps de réponse, etc. La propriété *produces*, quant à elle, permet de décrire la nature des données fournies par les capteurs, comme par exemple la température, la luminosité, la position, etc., figures 4.11 et 4.12. Le concept *Observation* est une représentation abstraite des données contextuelles fournies par les capteurs.

La propriété *observedBy* de l'ontologie *SSN* permet de lier sémantiquement le concept *Observation* aux sous-concepts décrivant les types de capteurs utilisés figures 4.12 . Le concept *Action*, qui est une notion fondamentale du modèle proposé, possède deux propriétés importantes : *hasAgent* et *controlsStateOf*, figure 4.13. La propriété *hasAgent* permet d'identifier l'objet actionneur permettant d'exécuter une action suite à un changement de contexte. La propriété *controlsStateOf*, quant à elle, permet d'identifier les objets de cette action.

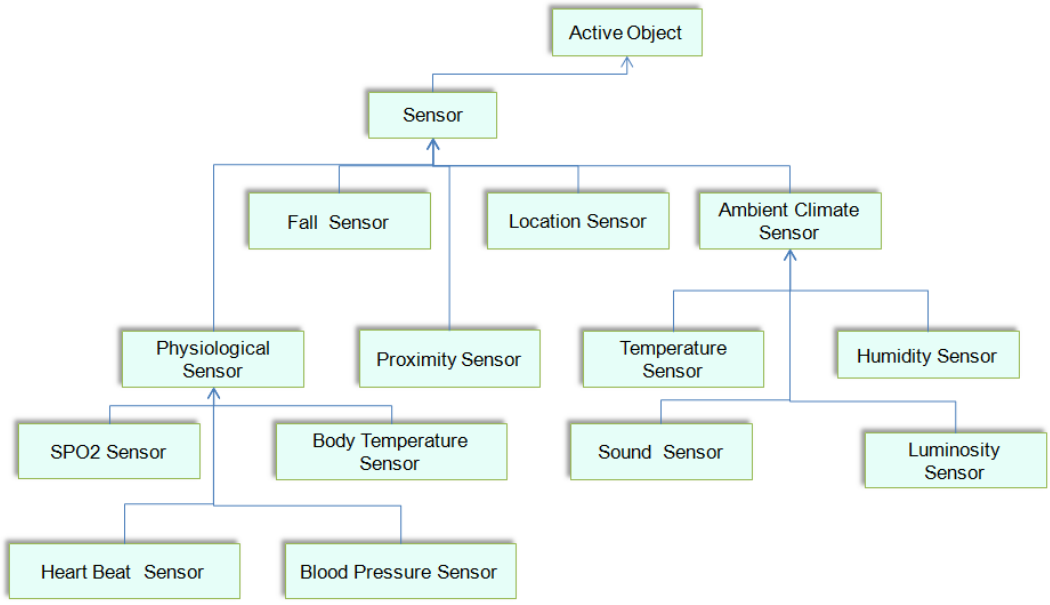

FIGURE 4.11 – Description des principaux capteurs.

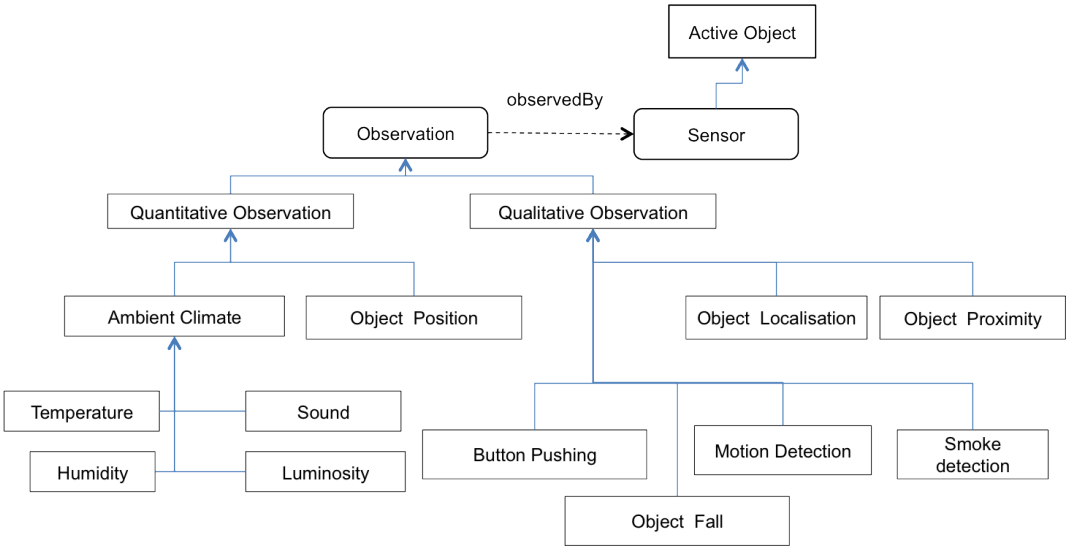

FIGURE 4.12 – Description des observations.

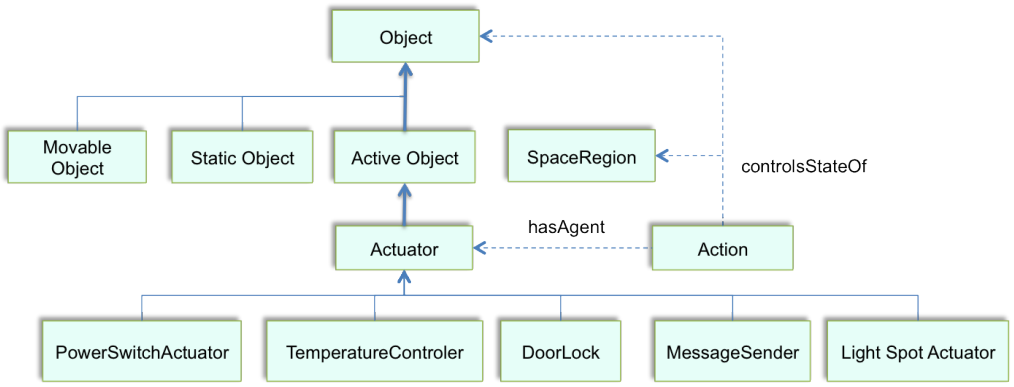

FIGURE 4.13 – Description des principaux actionneurs.

4.5 Règles de gestion de contexte

Dans notre approche, nous proposons des modèles de règles permettant d'une part, la reconnaissance de contextes implicites à partir de connaissances contextuelles explicites et d'autre part, l'adaptation au contexte. Nous distinguons deux types de règles. Le premier concerne les règles permettant de transformer des données fournies par les capteurs (Sensor_Ouput) en représentations sémantiques de haut niveau. Le deuxième type de règles permet d'agréger des contextes pour en déduire d'autres et déclencher des actions d'adaptation au contexte. Nous parlons dans ce cas de règles de raisonnement réactif, étant donné que seules les observations courantes sont considérées, figure 4.14.

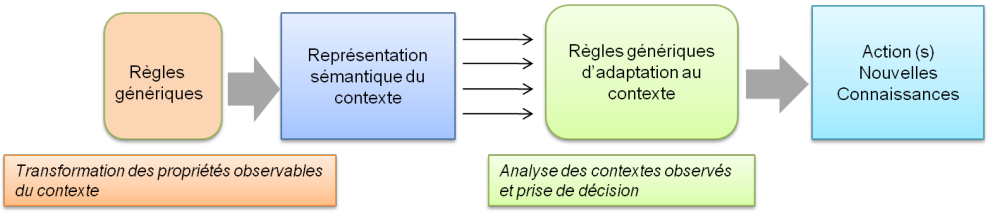

FIGURE 4.14 – Boucle de perception et d'adaptation au contexte.

Avant de présenter l'approche de raisonnement pour la reconnaissance de contexte, nous rappelons que contrairement au langage OWL, la représentation des connaissances en langage μConcept repose sur le principe du nom unique (Unique Name Assumption UNA). Cela signifie que deux symboles différents désignent deux entités différentes. Il est ainsi possible :

- de savoir si une instance donnée est déjà référencée, ce qui évite de créer un nombre infini d'instances et de propriétés associées à une même entité ;
- d'associer une instance référencée à sa représentation en langage μConcept.

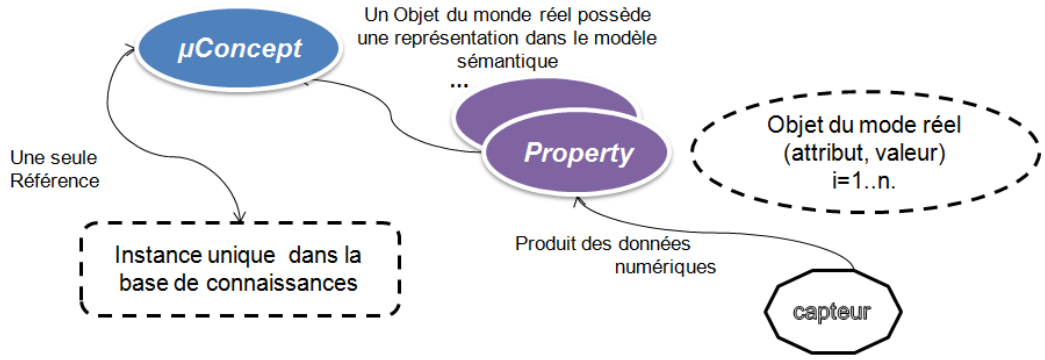

FIGURE 4.15 – Modélisation du passage numérique-symbolique des propriétés observables du contexte.

La figure 4.15 illustre la mise en œuvre du principe du nom unique où chaque entité de l'environnement est représentée sémantiquement par un seul concept dans l'ontologie *AmiOnt*, et par une seule instance dans la base de connaissances.

La notion d'identité en RDF et OWL

Dans le langage RDF, les ressources, les propriétés et les classes sont identifiées par des URIs. Cependant, si deux ressources ont deux URIs différentes, cela ne signifie pas que ces deux ressources soient différentes. Considérons par exemple le concept *Sensor* défini dans une ontologie OWL et une contrainte d'intégrité exprimant qu'un capteur de type *Sensor* ne peut être présent qu'à un seul endroit en même temps. À partir des trois triplets RDF suivants, nous ne pouvons pas exclure le fait qu'un lecteur RFID puisse se trouver à deux endroits en même temps, étant donné que rien n'atteste que les instances *ROOM1* et *ROOM2* sont différentes.

urn : RFID_READER :#101 rdf :typeSensor.
urn : RFID_READER :#101 ont :region Room1.
urn : RFID_READER :#101 ont :region Room2.

Pour pallier au problème ci-dessus, l'alternative proposée par le langage OWL est d'utiliser les propriétés *sameAs* ou *differentFrom*. La propriété *sameAs* peut être utilisée pour exprimer que des instances sont en réalité les mêmes. La propriété *differentFrom* exprime le fait que des instances du même concept sont différentes. Considérons, par exemple les deux instances *RFID_READER :#101* et *RFID_READER :#102* du concept *Identification*. Pour chaque radio-étiquette, une instance du concept *RFID_Tag* est créée. Par conséquent, compte tenu du principe de la non-supposition du nom unique en OWL, il est nécessaire d'ajouter à des axiomes définis au préalable dans une ontologie décrite en OWL, les deux assertions suivantes : i) expliciter que les instances associées à chacun des deux

lecteurs RFID sont identiques en utilisant la propriété *sameAs*, et ii) expliciter que les instances associées à ces deux lecteurs RFID sont différentes, à l'aide de la propriété *differentFrom*. En effet, sans ces deux assertions, les instances de concept *RFID_ Tag* associées à un même lecteur RFID peuvent être considérées comme différentes lors du processus du raisonnement.

Bien que les propriétés *differentFrom* et *sameAs* constituent une solution au problème soulevé ci-dessus, conserver toutes les instances associées à un capteur peut s'avérer coûteux en temps de traitement. En effet, il est nécessaire de vérifier, à chaque fois, que toutes les instances créées sont identiques ou différentes. La conservation des valeurs prises dans le temps par une instance peut engendrer des contradictions, et dans ce cas, la réaction du système peut devenir incohérente. Dans notre approche, seule la dernière valeur d'une propriété est conservée pour être utilisée par le processus de raisonnement.

Exemple : Étant donné que la localisation d'une entité est considérée comme connaissance contextuelle fondamentale pour la mise en œuvre l'adaptation au contexte, nous allons montrer comment exploiter le formalisme des règles *SmartRule* pour déduire l'espace où se trouve une personne. Considérons une personne habitant seule dans une maison composée de plusieurs espaces définis à partir de l'ontologie *AmiOnt*, tableau 4.2.

Concepts
Person, Robot, Hall, Indoor_ Space_ Region, Outdoor_ Space_ Region, House, Room, Linving_ Room, etc.
Relations
hasTag, locatedAt, allLocation, isOccupied, isOutSide, etc.
Actions a $\in T_A$
_ Open, _ SwitchOn, _ SendMessage, etc.
Contraintes $R \in T_R$
$Room \sqcap \leq 1 \ hasDoor$; etc. Cet axiome exprime qu'une chambre (*Room*) possède une seule porte.

TABLE 4.2 – Extrait des concepts, relations et actions

Dans ce scénario, nous supposons que la personne se trouve initialement à l'extérieur de sa maison. Cette connaissance contextuelle est représentée par la négation de la propriété *isAtHome* (a). Lorsque la présence de la personne est détectée dans l'un des espaces de la maison, les deux propriétés suivantes sont automatiquement positionnées à vrai : *isAtHome* et *isOccupied*.

(a) Person (not (isAtHome)) ;

(b) Space_Region (isOccupied) ; Person (isAtHome)

La mise à jour de ces deux connaissances (a) et (b) nécessite d'abord de considérer les instances liées aux concepts *Person et Object_ Localisation* puis de procéder à la mise à jour de leurs propriétés *isAtHome (e) et isOccupied (f).*

(c) ?peson := Person() ;

(d) ?space := Space_Region() ;

(e) ?person -> isAtHome := true ;

(f) ?space -> isOccupied := true ;

La règle *SmartRules* pour la localisation d'une personne dans un espace s'écrit :

```
rule "Localisation"
conditions
    Object_Localisation( ?observer := observedBy, ?space :=hasLocation) ;
    ?person :=Person() ;
actions
    ?person->locatedAt := ?space ;
    ?person -> isAtHome := true ;
    update( ?person) ;
end
```

Cette règle permet de déduire que la variable *?person* associée au concept *Person* se trouve dans un espace donné, représenté par la variable *?space* associée au concept *Space_ Region* à l'aide de la propriété *hasLocation.* La règle "Localisation" est déclenchée à chaque fois que la présence d'une personne est détectée dans un nouvel espace. Par conséquent, cette règle fournie une connaissance unique correspondant à l'espace où se trouve actuellement la personne. Ce nouvel état de l'environnement, caractérisé par une nouvelle connaissance (la personne se trouve chez-elle), peut conduire au déclenchement de règles d'adaptation au contexte, comme par exemple, éclairer le hall d'entrée quand la personne rentre chez-elle.

L'exemple ci-dessus montre aussi comment la règle générique "Localisation", permet de prendre en compte dynamiquement le changement des propriétés d'une instance d'un concept décrivant explicitement le contexte courant. Contrairement aux approches ontologiques du Web sémantique, l'approche de raisonnement non-monotone adoptée ici, grâce au principe de négation par l'échec permet de déduire de manière simple et non-erronée qu'une personne ne se trouvant pas à l'intérieur de son habitation se trouve forcément à l'extérieur, sans recourir à de nouvelles règles ou à des propriétés supplémentaires du type *isNotAtHome* et *isOutsideHome.*

4.6 Conclusion

Dans ce chapitre, nous avons développé une approche de modélisation sémantique et de raisonnement réactif fondée sur l'hypothèse du monde fermé avec supposition du nom unique pour la gestion du contexte dans les environnements intelligents ambiants et en robotique ubiquitaire. La proposition du langage μConcept répond en grande partie aux limites d'utilisation du langage OWL dans les applications nécessitant d'une part un raisonnement réactif non-monotone et d'autre part, une représentation d'instances de concepts et d'actions selon la supposition du nom unique. Pour compléter le modèle proposé, nous avons proposé l'ontologie *AmiOnt* composée de concepts de sens commun permettant de décrire de façon générique et expressive toutes les entités peuplant un environnement intelligent ambiant, et toute information, action ou événement pouvant être observée à partir de capteurs logiques ou physiques. Ces derniers sont définis en réutilisant des concepts proposés dans les ontologies W3C *SSN, DOLCE Ultra Lite (DUL)* et *NKRL HClass*.

L'ontologie *AmiOnt* constitue un support pour établir des règles définies au moyen du langage de règles *SmartRules* pour la gestion de la sensibilité au contexte qu'il s'agisse de règles pour la reconnaissance du contexte ou de règles pour l'adaptation au contexte. Les principales caractéristiques du modèle sémantique proposé sont sa souplesse et son extensibilité. Sans remettre en question les axiomes décrits dans l'ontologie *AmiOnt*, il est possible de mettre à jour ou d'étendre des modèles de contexte par simple modification, suppression ou ajout de règles représentées en langage *SmartRules*.

Chapitre 5

Modélisation sémantique et raisonnement narratif

Sommaire

5.1 Introduction

Dans ce chapitre, nous présentons le modèle de représentation de connaissances et de raisonnement NKRL (Narrative Knowledge Representation Language) pour la reconnaissance de contextes, à partir de l'historique des observations. Nous donnons tout d'abord les définitions utilisées tout au long de ce chapitre, puis nous présentons d'une part, les fondements de la modélisation des connaissances narratives à partir des ontologies HClass et Htemp et d'autre part, les mécanismes de raisonnement NKRL. Enfin, dans la dernière partie, nous développons une méthodologie d'annotation des événements élémentaires et dynamiques pour des applications en intelligence ambiante et en robotique ubiquitaire.

5.2 Les fondements de NKRL

Dans ce qui suit, nous définissons les principales notions utilisées dans le langage NKRL :

1. Événement narratif : On distingue deux types d'événements narratifs : Le narratif fictionnel et le narratif non-fictionnel (factuel). Le premier permet de créer une représentation des événements d'un monde imaginaire (simulation d'événements) ; le second sert, quant à lui, à représenter des événements se produisant dans le monde réel [82].

   Selon la vision de *Bal*, un événement narratif fournit la théorie classique de la narratologie [11]. L'auteur définit la narratologie comme une structure ternaire constituée de trois entités : La fable (*Fabula* en latin), l'*histoire* (*story*) et le *narratif* (*narrative*). L'entité *fabula* est vue comme un enchainement logique et chronologique des événements. L'entité *histoire* est un sous-ensemble particulier de *fabula* réarrangées en une nouvelle séquence. Enfin, l'entité *narratif* concerne la manière dont les événements sont racontés dans un langage ou média donné (image, vidéo, film).

2. Evénement élémentaire : Un événement élémentaire est considéré comme la manifestation d'attitudes spécifiques à l'égard d'êtres vivants ou d'objets abstraits. Il peut avoir une dimension spatiale et/ou temporelle, comme l'illustrent les exemples suivants : "déplacer une chaise", "envoyer un sms le soir", "mettre en marche la machine à café tôt le matin".

3. Entité statique : Elle représente un objet physique matériel (*wall, ceilling, tap,* etc.) ou immatériel qui n'est pas sujet à changement pendant la durée de vie de l'application.

4. Événement dynamique : Il représente une séquence d'événements élémentaires décrivant le comportement d'une entité (personne, objet), comme par exemple : Une personne actionne sa télécommande pour ouvrir une porte, le système déplace le robot suite à la chute d'une personne, etc. ;

5. Concept : Un concept NKRL peut être assimilé à la notion de classe en web sémantique. Il est associé à un ensemble de propriétés ou d'attributs. NKRL distingue deux catégories de concepts : Les concepts instanciables directement (*sortal_ concept*) et les concepts qui ne peuvent pas être instanciés directement (*non_ sortal_ concept*).

6. Concept instanciable : Par exemple, les instances *CHAIR_ 125, BED_ 2012, TAP_ 45*, etc. sont créées à partir des concepts *chair_ , bed_ , tap_* .

7. Concept non instanciable directement : Si nous considérons, par exemple, le concept *Couleur*, ce dernier ne peut pas avoir une instance directe. En effet, *RED_ 120* et *YELLOW_ 1* ne peuvent pas être considérées comme des instances puisqu'elles n'ont aucun sens si elles sont utilisées séparément. La solution apportée par NKRL consiste à introduire le concept *color_ appearance*, une spécialisation du concept instanciable *physical_ appearance*. Ainsi, il est

possible d'associer, une couleur à un concept instanciable, comme par exemple, *RED_ TABLE, YELLOW_ CUP*, etc.

8. Prédicat : NKRL n'est pas un formalisme universel, mais plutôt un formalisme permettant de représenter des événements non-fictionnels. Il consiste en sept prédicats de base pour déterminer la catégorie spécifique d'un événement, et identifier le type d'action, d'état, de situation, etc., tableau 5.1.

Prédicat	Description
BEHAVE	Ce prédicat permet de représenter des connaissances du type : Une personne ou un objet physique abstrait ou concret adopte une attitude particulière, joue un rôle particulier (concrètement ou intentionnellement), pour obtenir un résultat donné, comme par exemple : Une personne prend son repas, un groupe de personnes discutent ensemble, etc.
EXIST	Ce prédicat sert à représenter le fait qu'une entité peut être présente dans un espace donné. Par exemple, une personne est dans le séjour, une personne est décédée à l'hôpital.
EXPERIENCE	Ce prédicat traduit le fait qu'une entité est affectée par des événements. Par exemple, constater la diminution de l'intensité de la lumière dans un espace donné, une personne souffre d'un malaise, etc.
MOVE	À l'aide de ce prédicat, il est possible de représenter des faits liés au déplacement d'une entité, comme par exemple, déplacer un robot, envoyer un sms, etc.
PRODUCE	Ce prédicat permet de représenter l'exécution d'une tâche ou d'une activité par une entité. Par exemple, un capteur de température produit des informations liées à la température d'un espace donné.
OWN	Ce prédicat sert à représenter la notion de possession entre entités. Par exemple, la maison appartient à une personne, l'espace où la personne a chuté fait partie de la maison, une porte est fermée, etc.
RECEIVE	Ce prédicat permet de représenter des événements liés à la réception d'informations. Par exemple, une personne reçoit un appel téléphonique.

TABLE 5.1 – Prédicats de base NKRL.

9. Rôle : NKRL définit sept rôles conceptuels (en Anglais : conceptual roles) : SUBJECT, OBJECT, SOURCE, BENIFICIARY, MODAL, TOPIC et CONTEXT. Un rôle permet :
   - de représenter sémantiquement une relation fonctionnelle entre un prédicat et ses arguments.
   - d'identifier les acteurs d'un événement.

NKRL (Narrative Knowledge Representation Language) a été conçu à l'origine pour formaliser, gérer et traiter les principales connaissances décrites en langage naturel et contenues dans des documents principalement non-fictionnels tels que des rapports, des mémos, des enregistrements médicaux, etc. Le traitement des connaissances narratives en NKRL consiste à trouver une représentation suffisamment expressive pour décrire d'une part, des événements élémentaires et d'autre

part, les relations temporelles liant ces événements. La représentation conceptuelle de portions de connaissances narratives s'effectue grâce à l'ontologie de concepts HClass. Cette dernière s'apparente aux langages traditionnels d'ontologies tels que OWL, DAML+OIL. Pour la représentation des événements dynamiques, NKRL utilise une deuxième ontologie appelée HTemp. Cette ontologie constitue une solution bien adaptée aux problèmes de représentation de relations n-aires. NKRL s'appuie également sur des mécanismes d'inférence permettant d'établir automatiquement des relations sémantiques implicites ou explicites entre les connaissances. Il offre d'une part, un moyen de combiner les prédicats et les rôles conceptuels pour encoder de manière adéquate les informations narratives, et d'autre part, un support pour construire les liens sémantiques (causalité, but, etc.) entre événements élémentaires.

5.3 La représentation des connaissances narratives en NKRL

La représentation conceptuelle de connaissances narratives s'appuie sur quatre composants :

1. **Composant Définitionnel** *"definitional component "* : Ce composant a pour rôle la représentation des concepts, tels qu'un *bureau, maison, robot, capteur,* etc. Un concept peut être général tel qu'un être humain (*human_ being*) ou spécifique, comme par exemple, une *chaise*. Notons ici, que le concept *human_ being* est équivalent au concept *Person* défini dans l'ontologie *AmiOnt*. Dans les règles de nommage de NKRL, l'identifiant d'un concept est une chaîne de caractères en minuscules qui se termine par le caractère de soulignement.
2. **Composant énumératif** *"enumerative component"* : Ce composant constitue une énumération des instances de tous les concepts définis dans le composant définitionnel. Les règles de nommage pour les individus NKRL sont les mêmes que pour les concepts. Par exemple, DAVID_ et JOHN_ sont des instances du concept *human_ being* et *ROBOT_ KOMPAI* est une instance du concept *robot_* .
3. **Composant descriptif** *"descriptive component "* : Ce composant permet la représentation des structures (templates ou gabarits) d'événements élémentaires. Un événement élémentaire dans NKRL est constitué d'un prédicat, d'un ou plusieurs rôles et à chaque rôle peuvent être associés des arguments. Le tableau 5.2 représente la structure générale d'un template, où :

   - PREDICATE : Représente l'un des septs prédicats de base (PRODUCE, MOVE, EXPERIENCE, EXIST, OWN, RECEIVE, BEHAVE).

   - Argument : Représente les attributs pouvant être associés à chacun des rôles (SUBJ, OBJ, SOURCE, MODAL, TOPIC, CONTEXT, BENIFICIARY).

   - Location : Désigne l'espace où l'événement s'est produit.

   Enfin, *modulators* et *temporal attributes* sont des paramètres permettant la représentation temporelle des connaissances. Ces paramètres seront décrits

plus en détail dans le paragraphe suivant.

PREDICATE SUBJ { <argument > : [location] }
OBJ { <argument> : [location] }
SOURCE { <argument> : [location] }
BENF { <argument> : [location] }
MODAL { <argument> }
TOPIC { <argument> }
CONTEXT { <argument> }
[modulators]
[temporal attributes]

TABLE 5.2 – Structure générale d'un template NKRL.

4. **Composant factuel** *"factual component"* : Ce composant permet la représentation de toutes instances de tous les événements élémentaires. Une instance d'événement est appelée occurrence de prédicat *(en Anglais : predicative occurrence)*. L'instansiation d'une occurrence de prédicat obéit à la règle (5.1) suivante :

$$(L(P(R_1a_1)(R_2a_2).....(R_na_n))) \tag{5.1}$$

avec :

- L : Le label sémantique (symbole générique) permettant d'identifier de manière unique une occurrence de prédicat selon l'hypothèse du nom unique (UNA) ;

- P : Un prédicat NKRL ;

- R_k (k=1,...,n) : L'ensemble des rôles (SUBJECT, OBJECT, SOURCE, BENIFICIARY, MODAL, TOPIC et CONTEXT) ;

- a_k (k=1,...,n) : L'ensemble des arguments associés aux rôles.

En terme d'équivalence entre la représentation NKRL et la logique de description, les composants énumératif et définitionnel correspondent respectivement aux composants ABox et TBox. Cependant, il n'existe pas d'équivalents dans les langages de description pour les composants *descriptif* et *factuel*, figure 5.1.

Le tableau 5.3 décrit le template MOVE. Le symbole NLDescription permet de donner une description en langage naturel de ce template. Un rôle ou une variable définie entre crochets ([]) est considéré(e) comme optionnel(le). Dans le template MOVE, les rôles SUBJ, OBJ et les variables *var1 et var3* sont obligatoires, tandis que les rôles BENF, MODAL et CONTEXT, et les variables *var2, var4, var5, var6 et var7,* sont optionnelles. Les variables *var1, ... ,var7* représentent des contraintes permettant de vérifier que les valeurs assignées à chaque variable lors de la création d'une occurrence de prédicat sont spécifiques aux termes (concept, instances) utilisés dans le composant définitionnel. Ainsi, les contraintes définies dans les templates de l'ontologie HTemp sont associées aux concepts définis dans l'ontologie HClass. Par

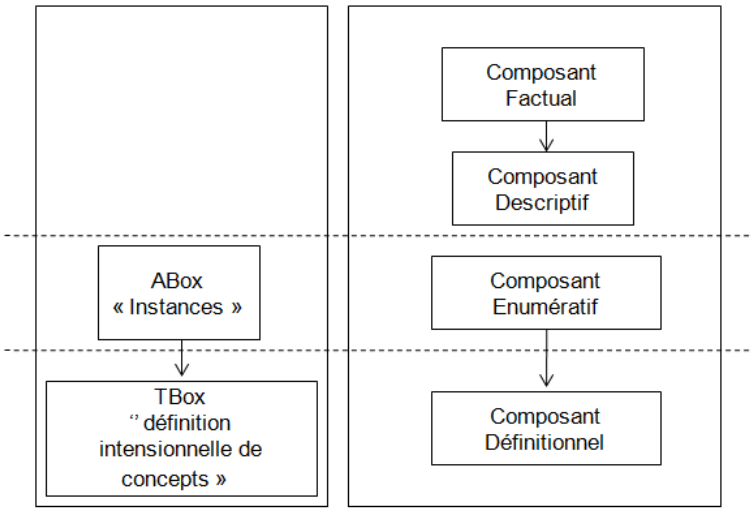

FIGURE 5.1 – NKRL vs Logique de Déscription.

conséquent, le processus de vérification de la consistance des connaissances s'appuie sur l'ontologie HClass pour établir une hiérarchie de concepts et d'instances basée sur le principe de généralisation/spécialisation.

```
NLDescription : 'Transmit a Structured Information'
PREDICATE : MOVE
                SUBJ var1 : [(var2)]
                OBJ var3
                [BENF    var4 : [(var5)]]
                [MODAL    var6]
                [CONTEXT    var7 ]

                {[modulators] !=abs}
                {[temporal attributes]}
var1 = <human_being> | <artefact_>
var2 = <location_>
var3 = <symbolic_label>
var4 = <human_being> | <artefact_>
var5 = <location_>
var6 = <media_> | <information_support>
var7 = <situation_>
```

TABLE 5.3 – Structure du template MOVE

5.3.1 Représentation des notions d'instant temporel et d'intervalle temporel en NKRL

La représentation temporelle des connaissances NKRL s'effectue par l'ajout d'annotations temporelles appelées également modulateurs temporels, permettant de

donner des précisions sur le début, la fin d'un événement, ou d'indiquer que cet événement a lieu à une certaine date, tableau 5.4.

Acronyme	**Description générale**
begin	représente le début d'un événement
end	marque la fin d'un événement
obs	utilisé en l'absence d'information temporelle sur le début ou la fin d'un événement. Par exemple, le système a observé que la température a commencé à augmenter à partir de 23h32.

TABLE 5.4 – Modulateurs temporels NKRL

Exemple 1 :

Supposons qu'une personne nommée John soit restée dans sa salle de bain pendant un intervalle de temps dont les dates de début et de fin sont respectivement : date-1=11/04/2011/7h :30mn et date-2=11/04/2011/8h :05mn. Le tableau 5.5 donne la représentation de l'occurrence du prédicat EXIST correspondant à cet événement. Cette occurrence a pour label (aal2.c2).

aal2.c2) EXIST SUBJ JOHN_ : (BATH_ROOM_1) date-1 : 11/4/2011/7 :30 date-2 : 11/4/2011/8 :05

TABLE 5.5 – Exemple d'occurrence du prédicat EXIST.

Exemple 2 :

L'occurrence (aal1.c4) du prédicat $EXPERIENCE$, représentée dans le tableau 5.6, indique l'augmentation (la propriété *growth_*) de la température à date-1= 11/08/2012/7h :35mn, mais ne donne pas d'indication sur la fin de cet événement. La propriété *temperature_* est utilisée pour expliciter qu'il s'agit d'une température, tandis que l'instance BATH_ROOM_1 représente l'espace où a lieu l'événement.

aal1.c14) EXPERIENCE SUBJ : HOME_CONTROL_SYSTEM_1 : (SPECIF temperature_ BATH_ROOM_1) OBJ : growth_ {*obs*} date-1 : 11/08/2012/7 :35 date-2 :

TABLE 5.6 – Exemple d'occurrence du prédicat EXPERIENCE.

Exemple 3 :

L'occurrence du prédicat (aal8.c42) ci-dessous est un exemple d'instance du prédicat PRODUCE, tableau 5.7. Cette occurrence représente l'événement où le système représenté par le symbole HOME_CONTROL_SYSTEM_1 a détecté (la propriété *detection_*) la présence d'une personne nommée John dans la cuisine (KITCHEN_1) à l'heure indiquée dans l'attribut temporel date-1.

aal8.c42) PRODUCE SUBJ HOME_CONTROL_SYSTEM_1 OBJ detection_ : (KITCHEN_1) TOPIC JOHN_ date-1 : 11/4/2011/16 :18 date-2 : Le 11/4/2011, à 16h18, le système détecte la présence de John dans la cuisine.

TABLE 5.7 – Exemple d'occurrence du prédicat PRODUCE.

5.3.2 Les structures de liaisons (Binding occurrence)

Ces structures permettent de définir des liens sémantiques entre événements élémentaires. Contrairement aux occurrences de prédicats, ces structures n'utilisent pas les symboles PREDICATE et ROLE pour exprimer des liens sémantiques.

La construction des structures de liaisons à partir d'opérateurs tels que COORD, GOAL, CAUSE, etc. suit la syntaxe suivante :

$$(oprateur \quad [arg_1 \quad arg_2 \quad ... \quad arg_n]) \tag{5.2}$$

où arg_1, ..., arg_i représentent des occurrences de prédicats.

Exemple. Supposons que John ait été détecté dans le couloir (HALL_1) et que le système éclaire cet espace. Les occurrences des prédicats MOVE et EXIST correspondant à ces deux connaissances sont respectivement :

aal.c6) MOVE SUBJ : HOME_CONTROL_SYSTEM_1
OBJ : lighting_ : (switch_off, switch_on)
TOPIC HALL_1
date-1 : 11/08/2012/22 :36
date-2 :

aal.c7) EXIST SUBJ : JOHN_ : (HALL_1)
date-1 : 11/08/2012/22 :34
date-2 :

L'opérateur de liaison CAUSE peut être utilisé pour lier ces deux occurrences, et exprimer que l'événement élémentaire représenté par l'occurrence (aal.c6) est provoqué par l'occurrence (aal.c7). Formellement,
(CAUSE aal.c7 aal.c6)

5.4 Les ontologies HClass et HTemp

La représentation des connaissances en NKRL s'appuie sur l'ontologie HClass (Hiérarchie de concepts) et l'ontologie HTemp (Hiérarchie d'événements). Les composants définitionnels et énumératifs sont définis dans l'ontologie HClass, alors que les composants descriptifs et factuels sont définis dans l'ontologie HTemp.

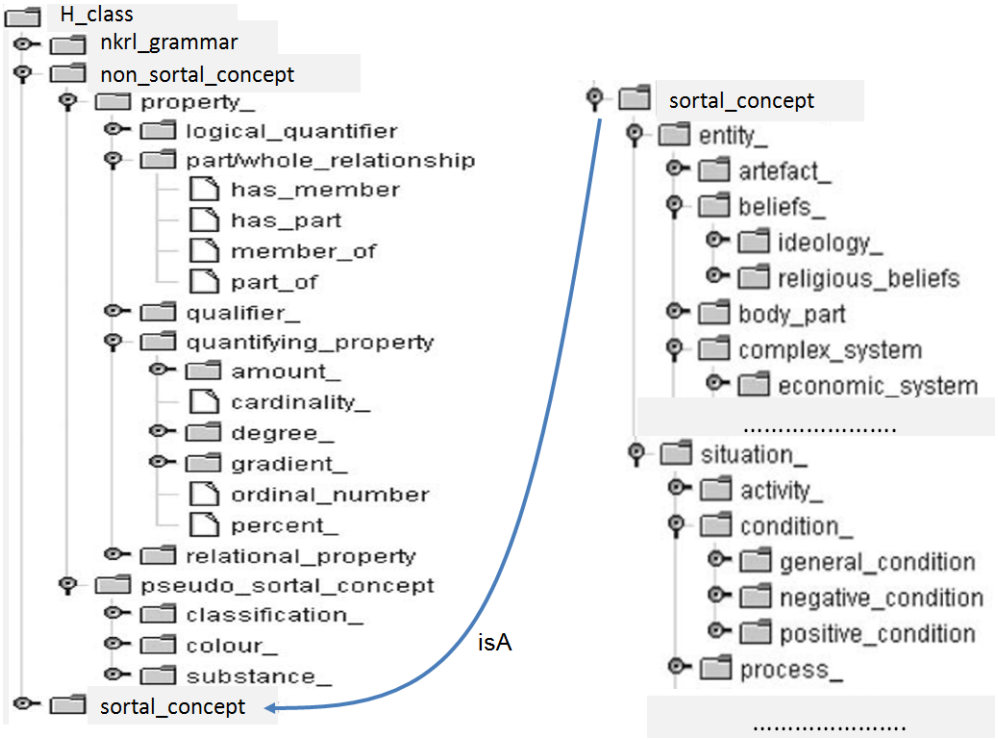

FIGURE 5.2 – Représentation arborescente de l'ontologie HClass

5.4.1 L'ontologie HClass

Elle est composée de trois branches : *nkrl_grammar, non_sortal_concept, sortal_concept*. Les concepts et instances qui composent ces trois branches sont organisés selon une taxonomie basée sur la relation de subsomption, figure 5.2. Les concepts subsumés par la branche *non_sortal_concept*, sont utilisés en général avec l'opérateur de spécification SPECIF. Ce dernier permet d'associer une liste de propriétés sous forme de concepts ou d'instances, aux rôles NKRL. Par exemple, la représentation conceptuelle de deux espaces différents (séjour et chambre) en NKRL s'écrit :

Exemple 1 :(SPECIF LIVING_ROOM_1 (SPECIF different_from ROOM_1))
La représentation conceptuelle d'un espace (couloir) situé ente deux espaces (salle de bain et séjour) est exprimée quant à elle en NKRL comme suit :

Exemple 2 : (SPECIF HALL_1 (SPECIF between _ BATH_ROOM_1 LIVING_ROOM_1))
Dans l'exemple 1, on remarquera l'utilisation d'une relation binaire entre les arguments LIVING_ROOM_1 et ROOM_1. L'exemple 2 illustre l'utilisation de la relation multiple *between_*, associant trois arguments : HALL_1, BATH_ROOM_1 et LIVING_ROOM_1.

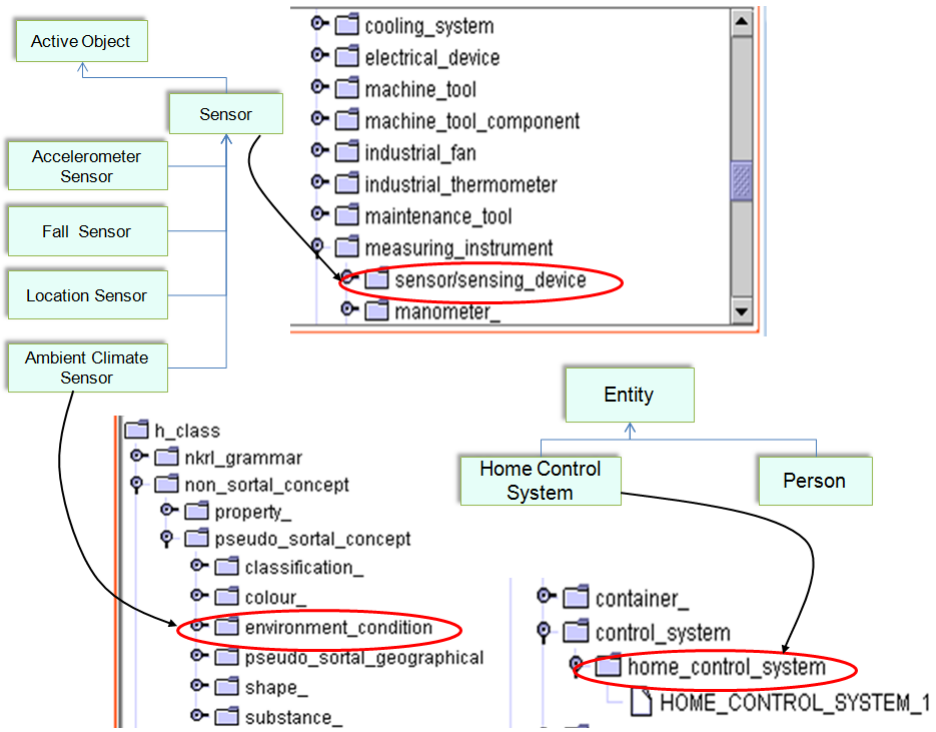

FIGURE 5.3 – Alignement de l'ontologie *AmiOnt* sur l'ontologie HClass

L'ontologie HClass qui comporte à l'origine plus de 2700 concepts a été étendue pour les besoins des applications d'intelligence ambiante développés dans le cadre de cette thèse. La figure 5.3 illustre l'alignement de l'ontologie *AmiOnt* sur les concepts HClass. Par exemple, le concept Sensor défini dans l'ontologie *AmiOnt* est équivalent au concept *sensor/sensing_ device* défini dans l'ontologie HClass.

5.4.2 L'ontologie HTemp

NKRL définit septs prédicats de base dans l'ontologie HTemp, figure 5.4

1. **BEHAVE** : Ce prédicat permet de représenter les actions effectuées par une ou plusieurs personnes ou les comportements de ces derniers. L'occurrence de prédicat ci-dessous représente le fait que la personne désignée par INDIVIDUAL_PERSON_108 est un voisin de John depuis date-1=11/4/2011 :

BEHAVE SUBJ INDIVIDUAL_PERSON_108
MODAL (SPECIF neighbour_ JOHN_)
{ obs }
date-1 : 11/4/2011/
date-2 :

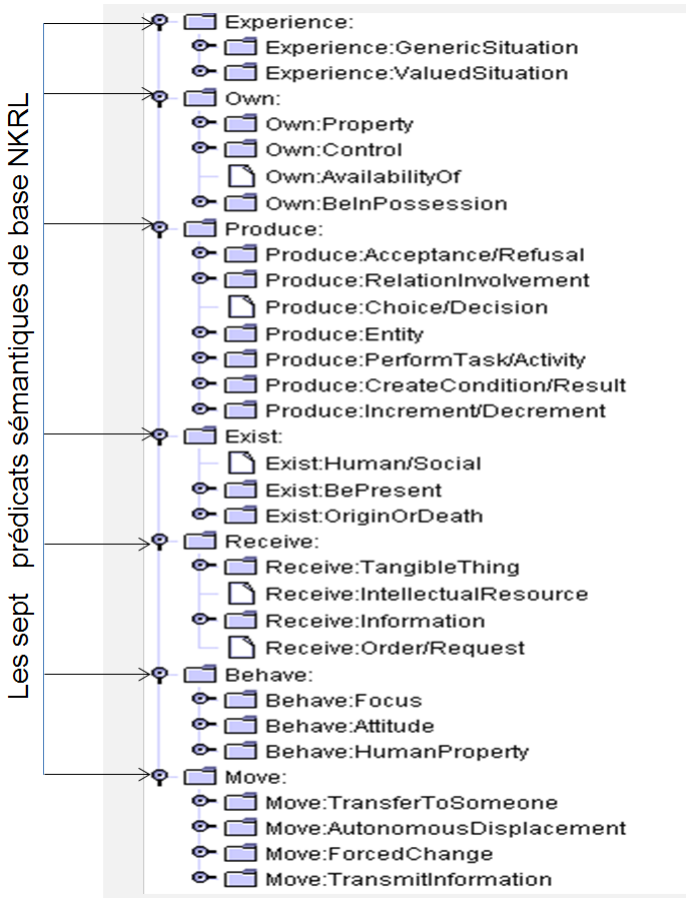

FIGURE 5.4 – Représentation arborescente de l'ontologie HTemp

2. **EXIST :** Ce prédicat est utilisé pour représenter un événement indiquant la présence d'une entité dans un espace donné.

   La représentation en NKRL de l'énoncé : John est couché sur le sol dans le séjour à une date-1=11/4/2011/17 :46, est comme suit :

   EXIST SUBJ JOHN_ : (LIVING_ROOM_1)
   MODAL lying_position
   { obs }
   date-1 : 11/4/2011/17 :46
   date-2 :

3. **EXPERIENCE :** Ce prédicat est principalement utilisé pour représenter un événement où une entité est affectée par un événement ou une expérience (la maladie, le succès, accident, etc.).

   L'occurrence de prédicat ci-dessous représente le fait que John souffre d'une insuffisance respiratoire depuis date-1=11/4/2011/17 :57

EXPERIENCE SUBJ JOHN_
OBJ respiratory_distress
{ obs }
date-1 : 11/4/2011/17 :57
date-2 :

4. **MOVE :** Ce prédicat sert à représenter une action de type déplacer, envoyer, etc.
   L'énoncé suivant : Le système a déplacé le robot de la position LOCATION_12 à la position LOCATION_13 à la date-1=11/4/2011/11 :37, est représenté à l'aide de l'occurrence de prédicat ci-dessous :

   MOVE SUBJ HOME_CONTROL_SYSTEM_1
   OBJ ROBOT_$KOMPAI$: (LOCATION_12, LOCATION_13)
   date-1 : 11/4/2011/11 :37
   date-2 :

5. **OWN :** Ce type de prédicat peut être utilisé pour représenter la notion de type possession entre entités ou l'état d'une entité. Ainsi, pour représenter en NKRL, le fait que la porte d'entrée est déverrouillée depuis date-1=11/4/2011/18 :06, on écrit :

   OWN SUBJ FRONT_DOOR_1
   OBJ property_
   TOPIC unlocked_
   { obs }
   date-1 : 11/4/2011/18 :06
   date-2 :

6. **PRODUCE :** Ce prédicat sert à représenter l'exécution d'une tâche, d'une activité, etc.
   L'occurrence du prédicat PRODUCE ci-dessous représente le fait que John est assis sur la chaise roulante WHEELCHAIR_1.

   PRODUCE SUBJ JOHN_
   OBJ sitting_activity
   TOPIC WHEELCHAIR_1
   date-1 : 11/4/2011/12 :03
   date-2 :

7. **RECEIVE :** Ce prédicat permet de représenter des événements liés à la réception d'information.
   À l'aide du prédicat RECEIVE, la représentation de l'énoncé : John a reçu un appel téléphonique dans le séjour à date-1=11/4/2011/12 :06, s'écrit :

RECEIVE SUBJ JOHN_ : (LIVING_ROOM_1)
OBJ (SPECIF information_content PHONE_CALL_2)
SOURCE HOME_CONTROL_SYSTEM_1
date-1 : 11/4/2011/12 :06
date-2 :

5.5 Mécanismes d'appariement et de représentation des règles

Les mécanismes de raisonnement NKRL s'appuient sur le principe question(requête)-réponse. Le traitement d'une question (requête) s'effectue via un ensemble de modules : Le module d'appariement (*Filter Unification Module (FUM))* et le module d'inférence des règles d'hypothèses et des règles de transformations.

5.5.1 Module d'appariement : Filter Unification Module (FUM)

Ce module utilise l'ontologie HClass pour apparier (unifier) des occurrences de prédicats de la base de connaissances avec une requête écrite sous forme d'une occurrence de prédicat. Cette opération est également basée sur le principe de généralisation/spécialisation entre concepts/instances définis dans l'ontologie HClass. Toute occurrence de prédicat qui est appariée avec la requête constitue alors une réponse plausible.

5.5.2 Règle d'hypothèse-règle de transformation

Une règle d'hypothèse est composée d'une prémisse et d'une ou plusieurs conditions. Chaque condition correspond à un pas de raisonnement, figure 5.5. Cet aspect est détaillé dans le paragraphe suivant. Une règle d'hypothèse permet d'expliquer les causes ayant conduit à la reconnaissance d'un contexte en reconstituant des liens sémantiques entre les événements. Il s'agit par exemple, de comprendre pourquoi un climatiseur a cessé de fonctionner durant un intervalle de temps donné, malgré la température élevée. L'idée sous-jacente ici est de prouver l'existence d'une occurrence de prédicat dans la base de connaissances indiquant qu'une personne a stoppé l'alimentation du climatiseur [142].

En cas d'impossibilité de trouver une réponse à une requête d'interrogation associée à une règle d'hypothèse, il est toujours possible d'obtenir une réponse à cette requête en la transformant en une nouvelle requête. Cette transformation consiste en une ou plusieurs règles de transformation.

Une règle de transformation est composée d'un antécédent qui correspond à une généralisation de la requête initiale créée à partir des conditions d'une règle d'hypothèse, et d'un ou plusieurs conséquents, figure 5.5.

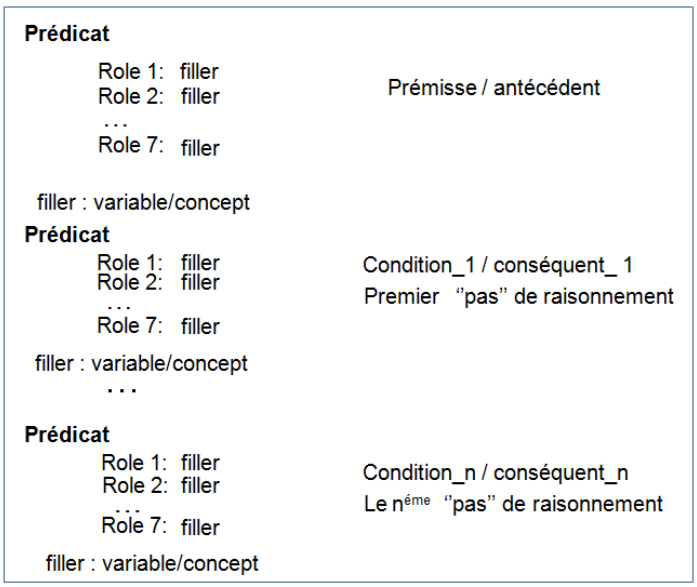

FIGURE 5.5 – Schéma général d'une règle d'hypothèse ou de transformation.

Une règle d'hypothèse ou de transformation est composée d'un prédicat, de rôle(s), d'attribut(s) et d'un ensemble de contraintes.

5.5.3 Raisonnement sur l'historique des observations

Avant d'exposer le mode de fonctionnement des procédures d'inférence NKRL, nous donnons la définition préliminaire suivante :

Definition 1 : Toutes les procédures d'inférence sont déclenchées selon la règle suivante :

$$X \quad iff \quad Y_1 \quad and \quad Y_2 \ ,,..., \quad Y_n. \tag{5.3}$$

où X représente le contexte à inférer, et Y_1..... Y_i, les étapes (ou pas de raisonnement) pour effectuer toutes les opérations d'appariement avec les occurrences de prédicats présentes dans la base de connaissances.

La figure 5.6 schématise le processus d'inférence pour la reconnaissance d'un contexte. Ce processus s'effectue comme suit :

À chaque assignation de contrainte à une variable *vari* de chaque prémisse/antécédent, une occurrence de prédicat est instanciée. Cette occurrence, appelée motif de recherche, représente une requête (question) à laquelle le moteur d'inférence tente de trouver un appariement (unification) avec les occurrences de prédicats présentes dans l'historique des observations. Chaque condition/conséquent d'une règle d'hypothèse ou de transformation correspond à un pas de raisonnement, symbolisé par Y_i, figure 5.6.

L'obtention d'une réponse à une requête (motif de recherche), signifie que la(e)

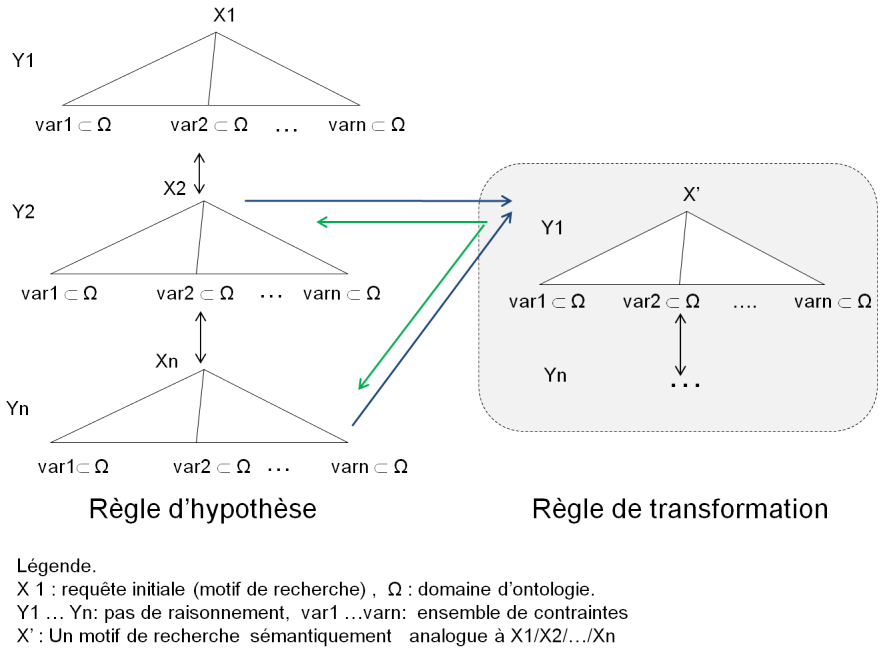

FIGURE 5.6 – Processus d'inférence NKRL.

condition/conséquent est satisfait(e). Le processus d'inférence se poursuit en traitant la requête correspondant à la(e) condition/conséquent suivant(e). L'interrogation successive de l'historique des observations se poursuit jusqu'à atteindre le dernier pas de raisonnement symbolisé par Yn, figure 5.6. Deux cas peuvent alors se présenter :

1- Dans le cas d'une règle de transformation, cela signifie que le motif de recherche construit à partir de la règle d'hypothèse peut être substitué par un autre motif de recherche X' pour poursuivre le traitement de la règle d'hypothèse.

2- Si le processus d'inférence concerne une règle d'hypothèse, cela signifie qu'une réponse plausible expliquant le contexte symbolisé par $X1$ a été déduite.

D'un point de vue implémentation, une requête d'interrogation $NKRL$ est une requête $SPARQL$ sur une base de connaissances OWL. Par exemple, la requête 1 ci-dessous permet de trouver tous les espaces (*location_*) où la personne a pu se trouver, tandis que la requête 2 restreint la recherche à un intervalle temporel défini par les bornes date-1 et date-2 :

Requête 1 : PREDICATE EXIST
SUBJ : human_being : location_

Requête 2 : PREDICATE EXIST SUBJ human_being
date-1 : 11/4/2011/11 :30
date-2 :11/4/2011/11 :40

5.5.4 Algorithme de raisonnement temporel

L'accès aux connaissances narratives stockées dans la base de connaissances s'effectue à partir d'un schéma d'indexation. Ce dernier combine trois types d'éléments :

1. Les prédicats associés à chaque occurrence de prédicat ;
2. Les arguments de chaque prédicat ;
3. Les attributs temporels (date-1 et date-2) associés à une occurrence de prédicat.

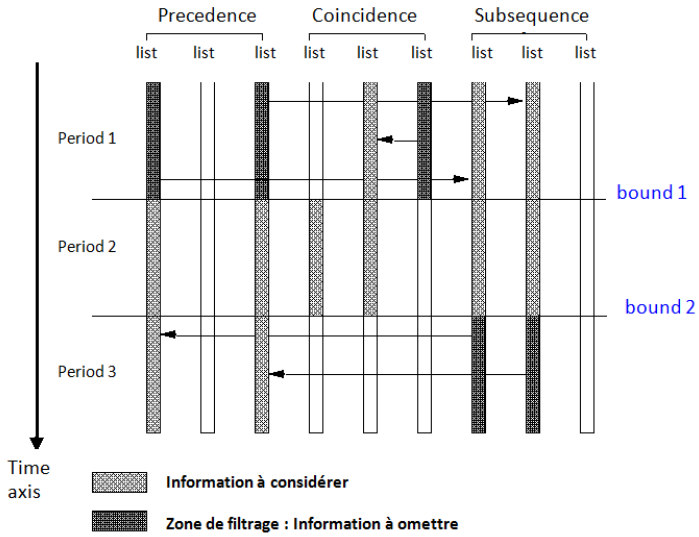

FIGURE 5.7 – Représentation graphique de l'algorithme d'indexation, adapté de [160]

Pour expliquer l'algorithme de raisonnement temporel sur lequel est fondé $NKRL$, considérons les deux exemples suivants, figure 5.7 :

Exemple 1 :
Une personne nommée John qui était en convalescence, est rentrée chez-elle à la date date-1= 25/04/2011. Cinq jours plus tard, John a été ré-hospitalisé. La représentation conceptuelle en $NKRL$ de la présence de John chez-lui à partir de la date date-2=25/04/2011, s'écrit :

aal1.c1) EXIST SUBJ JOHN_ :(HOME_1)
[begin]
date-1 : 25/04/2011/
date-2 :

Pour des besoins de suivi médical, on désire savoir si John était à l'hôpital entre le 29/04/2011 et 12/05/2011. Cette requête peut être exprimée en $NKRL$ à l'aide de la requête suivante :

[*query*1] = EXIST SUBJ JOHN_ :(HOSPITAL_12)
(29/04/2011, 12/5/2011)

Dans la requête *query1*, l'information temporelle (29/04/2011, 12/5/2011) permet d'exclure les occurrences de prédicats dont les dates ne sont pas incluses dans cet intervalle. En effet, chaque occurrence de prédicat est référencée à l'aide d'un couple d'attributs : Le label de l'occurrence et les attributs temporels : date-1 et date-2.

Dans la base de connaissances, toutes les dates sont organisées sous-forme de neuf listes correspondant aux trois catégories temporelles suivantes, , figure 5.7 : *precedence, coincidence, subsequence*. La catégorie *precedence* représente les événements apparus avant la date indiquée dans l'attribut *date-1*. La catégorie *subsequence* représente les événements qui se sont produits après *date-2*. La catégorie *coincidence* permet, quant à elle, de représenter les événements utilisant le modulateur *obs*. Ces trois catégories se composent chacune de trois listes. Chaque liste est séparée en trois sections qui correspondent à *period 1, period 2* et *period 3*. Les bornes *bound 1 et bound 2* délimitent ces périodes. Dans notre exemple, *bound 1* correspond à la première borne (29/04/2011) de l'intervalle (29/04/2011, 12/5/2011), et *bound 2* à la seconde borne (12/5/2011) de l'intervalle (29/04/2011, 12/5/2011). Ainsi, *period 1* correspond à la période avant la borne *bound 1*, *period 2* à la période entre les deux bornes *(bound1 et bound 2)*, et *period 3* à la période après la borne *bound 2*.

Dans l'occurrence de prédicat (aal1.c1) donnée en exemple, l'information temporelle permet de ne sélectionner que les occurrences de prédicats dont la date correspond à la seconde borne *bound 2* (toutes les occurrences de prédicats dont l'attribut date-1 appartenant à la catégorie *concidence* sont sélectionnées). Les occurrences de prédicats dont la date est incluse dans la période *period 1* sont exclues.

5.6 Processus d'annotation des informations narratives en NKRL

Avant de présenter notre proposition d'annotation des événements élémentaires et dynamiques, nous exposons de manière détaillée la méthodologie d'extraction des informations narratives et leurs encodages en NKRL. Ce dernier a été conçu à la base pour le traitement d'informations narratives contenues dans des documents. Ainsi, l'encodage sous forme d'occurrences de prédicats de ces informations consiste à :

1- Extraire toutes les informations intéressantes existant dans des documents, dépêches, etc.

2- Éliminer les informations éventuellement redondantes. En effet, la présence de ces informations ralentirait le moteur d'inférence lors de l'exploration de la base de connaissances.

Une fois ces deux étapes terminées, la dernière étape consiste à construire les occurrences de prédicats en sélectionnant le prédicat le plus adéquat à chaque information.

5.6.1 Choix du Prédicat

La sélection d'un template dans l'ontologie HTemp revient à sélectionner le prédicat NKRL (PRODUCE, MOVE, BEHAVE, EXIST, EXPERIENCE, OWN, RECEIVE) le mieux adapté sémantiquement pour représenter un événement élémentaire. Les prédicats NKRL sont basés sur les théories Davidsonienne et néo-Davidsonienne pour la définition des événements [32].

Dans ce qui suit, nous rappelons brièvement le principe de ces théories dans le domaine de la linguistique en s'inspirant de l'analyse détaillée de *Landamn* [66].

Considérons les énoncés suivants :
- (EN1) David allume la télévision.
- (EN2) David allume la télévision dans le séjour à l'aide de la télé-commande.

Dans l'analyse détaillée de *Landman*, les théories Davidsonienne et néo-Davidsonienne dans le domaine linguistique considèrent qu'un verbe comme *allumer* exprime une relation entre deux arguments et les adverbes agissent comme des modifieurs de verbe. Autrement dit, un adverbe est une sorte de fonction qui a pour entrée un verbe et renvoi un autre verbe en retour [92] [137]. En revanche, pour *Davidson* [32], un verbe exprimant l'action (comme allumer) utilise trois arguments : Deux arguments *nominaux* et un argument *événement* (e). Ce dernier est implicitement quantifié à l'aide du quantificateur existentiel "$\exists$". La théorie de *Davidson* propose de définir des adverbes comme prédicats de l'argument *événement*, et par conséquent, les deux énoncés (EN1) et (EN2) peuvent être représentés selon la théorie de *Davidson* comme suit :
- $\exists$ e(Allumer(e,David,Télévision))
- $\exists$ e(Allumer(e,David,Télévision)) $\wedge$ Dans(Séjour,e) $\wedge$ Avec(Télécommande,e))

Où e est une variable représentant un événement.

Ainsi le sens global de l'énoncé (EN1) devient : Il existe un événement e tel que e représente la mise en marche de la télévision (se trouvant dans le séjour) par David.

Les néo-Davidsoniens comme *Higginbotham* [54] et *Parsons* [101] considèrent les verbes non pas comme des relations, mais plutôt comme des prédicats unaires, et que les adverbes et les arguments sont associés aux verbes via des rôles thématiques. Par conséquent, les énoncés (EN1) et (EN2) peuvent être représentés comme suit :
- $\exists$ e(Allumer(e) $\wedge$ Agent(David,e) $\wedge$ Thème(Télévision,e))
- $\exists$ (Allumer(e) $\wedge$ Agent(David,e) $\wedge$ Thème(Télévision,e) $\wedge$ Location(Séjour,e)$\wedge$ Instrument(Télécommande,e)).

Pour comprendre comment un prédicat NKRL est associé à un événement, consi-

dérons à nouveau l'énoncé (EN2). Le prédicat $MOVE$ est ici sélectionné car il est le plus adéquat. Une occurrence de ce prédicat est ainsi créée. Concrètement, le symbole DAVID_ représente l'attribut du rôle SUBJ. L'instance TELEVISION_ se trouvant dans le séjour est associée à l'attribut du rôle OBJ. La modalité associée à cette instance est représentée à l'aide du symbole TELECOMMANDE_1, tableau 5.8.

PREDICATE MOVE SUBJ DAVID_ OBJ (SPECIF TELEVISION_ SEJOUR_1) : (switch_off, switch_on) MODAL TELECOMMANDE_1

TABLE 5.8 – Exemple d'occurrence de l'énoncé EN2.

Dans le cadre de cette thèse, les étapes d'interprétation et d'encodage ne posent pas de problème en pratique. En effet, nous avons expliqué au début du mémoire que notre approche de gestion de contexte dans le cadre de l'intelligence ambiante est basée sur l'hypothèse du monde fermé. Par conséquent, les concepts et leurs relations sont définis à la conception et le raisonnement s'effectue sur des connaissances complètes. Par la maitrise de la sémantique de description de toutes les entités et de leurs relations, nous avons une interprétation unique du rôle de chaque entité de l'environnement et des événements associés.

Le choix d'un template dans l'ontologie HTemp consiste à sélectionner un des prédicats de base $NKRL$ (*PRODUCE, MOVE, BEHAVE, EXIST, EXPERIENCE, OWN, RECEIVE*) correspondant sémantiquement à un événement élémentaire.

La représentation conceptuelle d'évènements par des occurrences de prédicats nécessite la définition d'un modèle générique permettant de faire un appariement entre les templates $NKRL$ et un événement élémentaire observé dans le monde réel. Ce modèle est vu comme une fonction ayant comme paramètre d'entré un prédicat représentant l'évènement élémentaire, et comme valeur de retour un template $NKRL$. Ainsi, les relations entre les entités du monde réel et leurs représentations sémantiques sont définies par une table de correspondance prédéfinie. Cette table définit un ensemble d'éléments où chacun est associé à un template. Par exemple, un détecteur de présence est référencé par le prédicat $EXIST$. Compte tenu de l'hypothèse du monde fermé, tous les éléments de la table sont connus a priori. Dans notre méthodologie de création des occurrences de prédicats à partir des événements observés, nous excluons la représentation conceptuelle des structures de liaisons $NKRL$ ($COORD$, $ALTER$, etc.). De notre point de vue, l'utilisation de ces structures nécessite d'autres techniques algorithmiques qui sortent du cadre de cette thèse.

La figure 5.8 donne une vue simplifiée du module de transformation des informations contextuelles de bas niveau en connaissances contextuelles de haut niveau, représentées selon le modèle $NKRL$. Le module *Enveloppeur (Wrapper module)* a pour rôle d'encoder les valeurs numériques de chaque évènement élémentaire en un message au format XML, compréhensible par le module de gestion (*Manager*

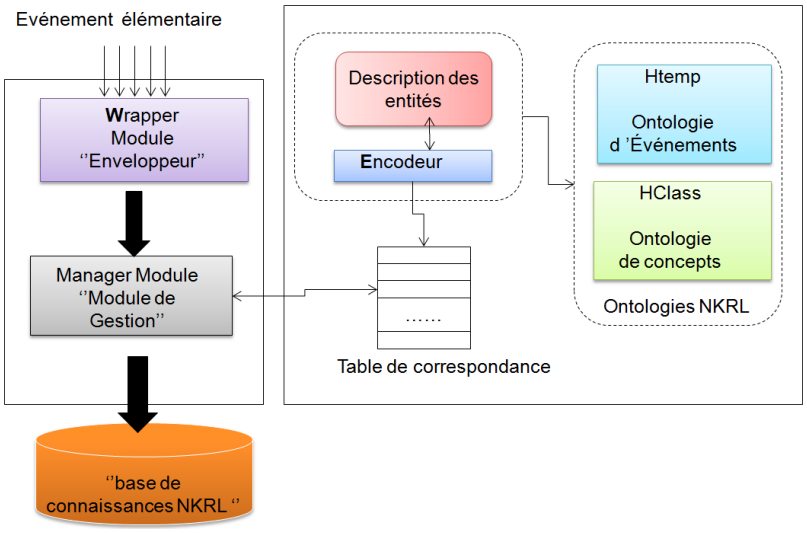

FIGURE 5.8 – Architecture de transformation d'un événement observé en occurrence de prédicat.

Module). Ce dernier agit en tant qu'orchestrateur et gère la communication entre les différents modules. Il est également responsable de l'insertion de nouvelles occurrences de prédicats dans la base de connaissances selon les étapes suivantes : i) extraction de l'ID de l'événement en cours de traitement, ii) consultation de la table de correspondance, et iii) création des occurrences de prédicats.

À titre d'exemple, considérons par exemple, un détecteur de présence installé dans le séjour ($LIVING\_ROOM$). Ce détecteur permet de détecter la présence d'une personne dans le Séjour à date-1=11/04/2011/17 :51.

À partir des descriptions de l'entité (détecteur de présence) et du prédicat $EXIST$, il est possible de représenter la présence d'une personne dans le séjour par une occurrence de prédicat. Notons ici que le rôle $SUBJ(ect)$ doit être défini. Si on s'intéresse uniquement au rôle $SUBJ(ject)$, le fichier XML doit comporter trois informations contextuelles : L'identité de la personne détectée, l'espace où elle se trouve et le temps associé à la détection de la présence de cette personne dans le séjour. L'identité de la personne détectée peut être obtenue en utilisant un *Tag RFID*.

5.7 Conclusion

Dans ce chapitre, nous avons présenté le modèle de représentation de connaissances et de raisonnement NKRL (Narrative Knowledge Representation Language). La représentation des connaissances en NKRL s'appuie sur l'ontologie HClass (Hié-

rarchie de concepts) et l'ontologie HTemp (Hiérarchie d'événements). Cette dernière est bien adaptée pour la représentation de relations n-aires. NKRL s'appuie sur des mécanismes d'inférence permettant d'établir des relations sémantiques implicites ou explicites entre les connaissances. Il permet de combiner les prédicats et les rôles conceptuels pour encoder de manière adéquate les informations narratives, et de construire les liens sémantiques (causalité, but, etc.) entre événements élémentaires. Le processus de raisonnement NKRL s'appuie sur le principe question (requête)-réponse et un mécanisme d'appariement, pour le traitement des règles d'hypothèses et de transformations. Pour l'exploitation de ce modèle, une approche d'annotation des événements élémentaires et dynamiques a été proposée. À travers ce chapitre, nous avons montré que la grande expressivité et le raisonnement narratif du modèle NKRL rend ce dernier particulièrement bien adapté pour la reconnaissance de contextes non-triviaux pouvant être liés au temps.

Chapitre 6

Mise en œuvre et validations expérimentales

Sommaire

6.1 Introduction

Ce chapitre est dédié à la mise en œuvre et à la validation expérimentale des modèles sémantiques, et des raisonnements réactif et narratif, proposés dans les chapitres 4 et 5 pour la gestion du contexte. Dans la première partie du chapitre, nous décrivons la plateforme ubiquitaire développée au laboratoire dans le cadre des ces travaux de thèse et des projets européens SEMBYSEM et WOO. Nous décrivons en particulier l'architecture logicielle d'implémentation pour la gestion du contexte. Dans la suite du chapitre, nous présentons l'implémentation de l'ontologie *AmiOnt* pour la mise en œuvre de scenarii d'assistance cognitive et de reconnaissance de contexte. Enfin, dans la dernière partie, nous montrons, à travers un scénario de gestion de situation d'urgence, l'apport du raisonnement narratif pour la construction de liens sémantiques entre événements et reconnaissance de contextes complexes non-observables.

6.2 Description de la plateforme ubiquitaire du LISSI

Cette plateforme a été mise en œuvre pour des besoins de validation des modèles de représentation de connaissances et de raisonnement développés dans les chapitres 4 et 5. Les scénarii exposés dans la suite de ce chapitre ont été implémentés en exploitant cette plateforme qui comprend différent composants :

- Un robot compagnon ;
- Des dispositifs d'affichage : Téléphone portable de type *Smartphone*, moniteur de *PC*, écran d'affichage du robot compagnon ;
- Un système de localisation indoor ;
- Un ensemble de capteurs permettant d'observer des évènements liés au contexte d'une personne âgée à domicile :
  1. Un lecteur $RFID$ longue porté (de 1 à 8 mètres) permettant d'identifier une entité portant un tag actif. Ce lecteur, embarqué sur le *robot* compagnon, permet aussi d'estimer la position grossière de l'entité en exploitant le système de localisation du *robot* ;
  2. Un lecteur $RFID$ courte portée (quelques centimètres), porté au poignet d'une personne, permet d'identifier une entité taguée et d'établir sa proximité par rapport à la personne ;
  3. Des capteurs de détection d'ouverture de portes, de détection de présence, de mesure d'intensité lumineuse et de température ;
  4. Un bracelet permettant de détecter la chute d'une personne, de mesurer son pouls et de remonter une alarme par appui sur un bouton d'urgence ;
  5. Un actionneur de type TOR (Tout Ou Rien) permettant d'actionner une prise de courant à distance, et ainsi de mettre en marche un équipement ;
- Des actionneurs dédiés permettant d'allumer/d'éteindre des ampoules électriques à distance ;

La figure 6.1 donne une vue partielle de la plateforme ubiquitaire. Dans ce qui suit, nous donnons une description synthétique de chacun des composants de cette plateforme.

6.2.1 Le robot compagnon Kompai

Le robot *Kompai* développé par la société *Robosoft*, est équipé de plusieurs capteurs et actionneurs permettant d'assurer des fonctions essentielles telles que la planification de trajectoire, la navigation en environnement encombré, la cartographie, la localisation, l'interaction pour des tâches d'assistance au quotidien, figure 6.2. Il est équipé de :

- Deux caméras ;
- Une tablette-PC ;
- Un télémètre laser ;
- De capteurs à ultrasons ;
- De capteurs à infrarouge ;

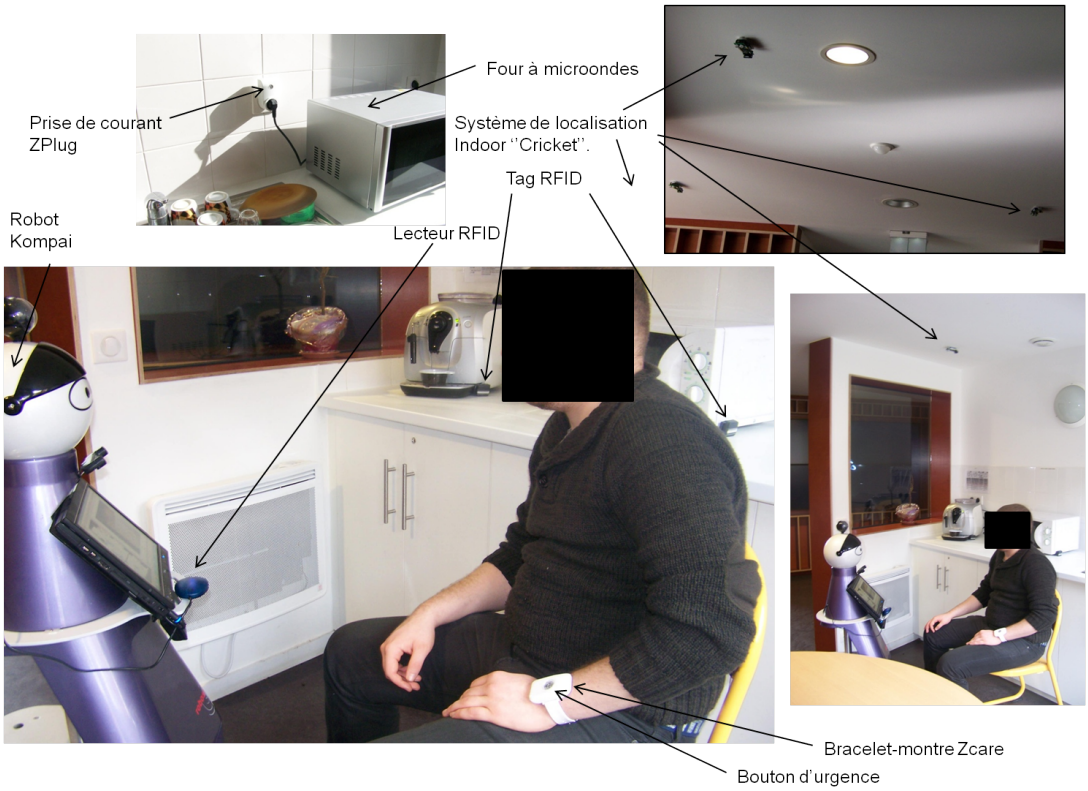

FIGURE 6.1 – Vue partielle de la plateforme ubiquitaire du Laboratoire LISSI

- De détecteurs de contact ;
- Deux moteurs ;
- Une batterie ;
- Une unité de contrôle embarquée ;
- Une interface Wifi ;

Le robot *Kompai* embarque également une importante partie logicielle qui assure des fonctions de contrôle de bas niveau, jusqu'aux services de haut niveau à destination de l'utilisateur final. Ainsi, le robot propose des services tels que l'agenda, la messagerie *Skype*, la gestion des mails, la possibilité de faire ses courses en ligne, le contrôle par reconnaissance vocale, etc.

L'architecture logicielle open source du robot, appelée *RobuBOX* est constituée de trois couches, figure 6.3 :

– RobuBOX Services : Elle permet d'ajouter des services de plus haut niveau en utilisant l'intergiciel Microsoft Robotics Developer Studio (MRDS). Comme exemples de services, on peut citer : Les services de cartographie, de navigation, de planification de trajectoire, etc ;
– RobuBOX Lokarria : Elle consiste en un ensemble de services fournissant un accès à distance au robot ;

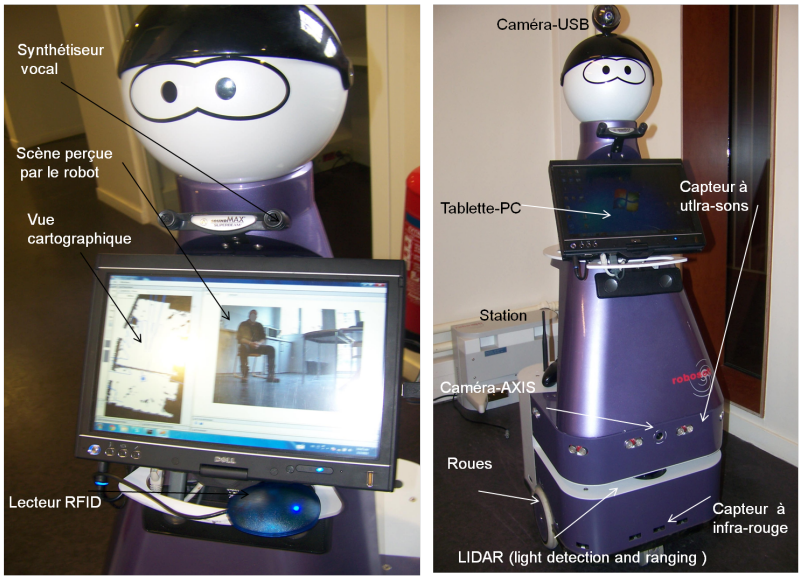

FIGURE 6.2 – Le robot Kompai.

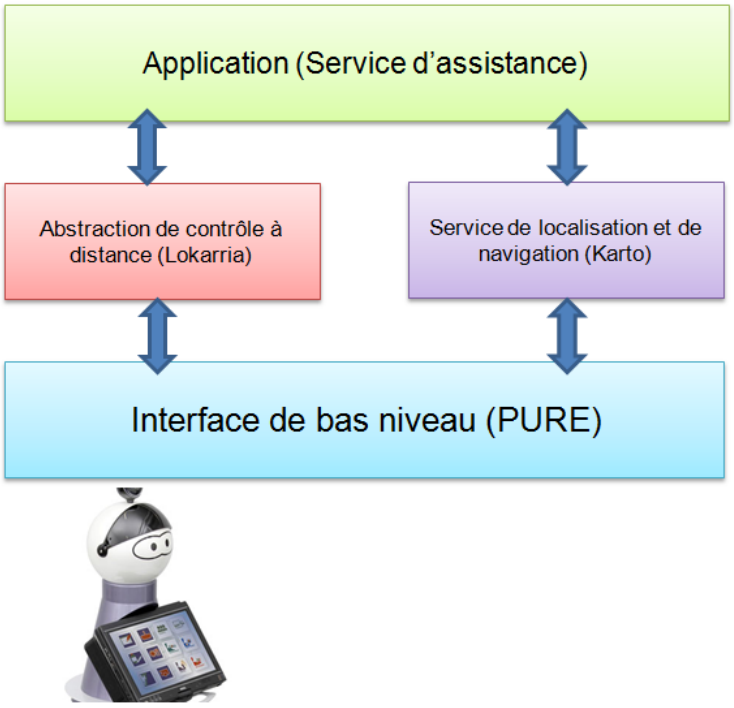

FIGURE 6.3 – Les trois couches de l'architecture robuBOX.

– RobuBOX PURE : Elle est chargée des contrôles de bas niveau (boucles de contrôle, machine à états, etc.) ;

Si la quasi-totalité de la robuBOX est codée en C#, de nombreux services disposent d'interfaces REST qui permettent le contrôle du robot par des applications tierces, en utilisant le protocole HTTP. Le robot *Kompai* intègre le logiciel *Karto* qui est une suite logicielle incorporant des algorithmes avancés pour la cartographie et la navigation autonome des robots mobiles. *Karto* offre les fonctionnalités suivantes : La localisation, la cartographie par technique de SLAM, l'exploration, la planification de trajectoires et l'évitement d'obstacles. Le logiciel Karto est livré avec un module réseau qui permet la transmission de données distantes (odométrie, balayage laser) vers le module de cartographie (serveur SLAM).

6.2.2 Système de localisation indoor

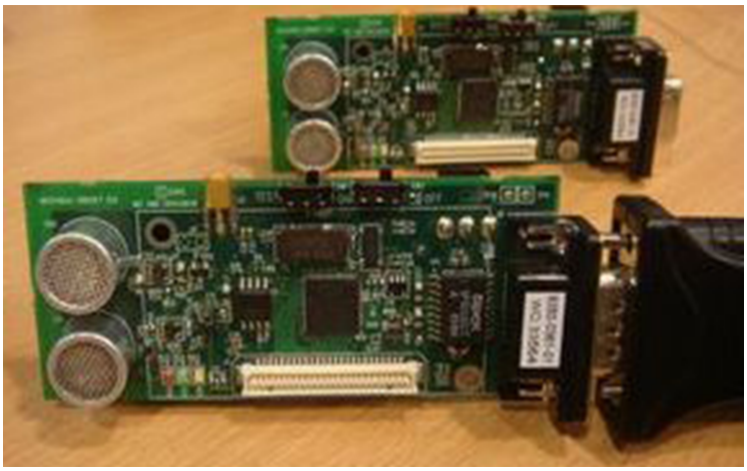

FIGURE 6.4 – Le système de localisation indoor "Cricket".

Cricket est un système de localisation indoor conçu à l'origine par le MIT, figure 6.4. Il s'agit d'un réseau de capteurs sans fil basse consommation qui fournit deux informations de localisation. La première représente l'identifiant de l'espace où se trouve l'entité à localiser : *Kitchen, Living_ Room, Room,* etc. La seconde information représente la position courante de l'entité en coordonnées cartésiennes 3D.

La façon la plus courante d'utiliser le système de localisation *Cricket*, consiste à déployer des balises de transmission (*Cricket beacons*) sur les murs ou les plafonds, et d'attacher la balise d'écoute (*Cricket listener*) à une entité mobile (*Personne, objet mobile*, etc.) dont la localisation doit être déterminée, figure 6.5. La position de l'entité mobile est ensuite estimée à partir de la mesure de la différence entre le temps de propagation des ondes RF et des ondes ultrasonores. À chaque fois que la balise d'écoute intercepte une information des balises de transmission, elle infère les coordonnées de sa position en se basant sur les distances par rapport aux balises de transmission (dont les positions sont connues a priori).

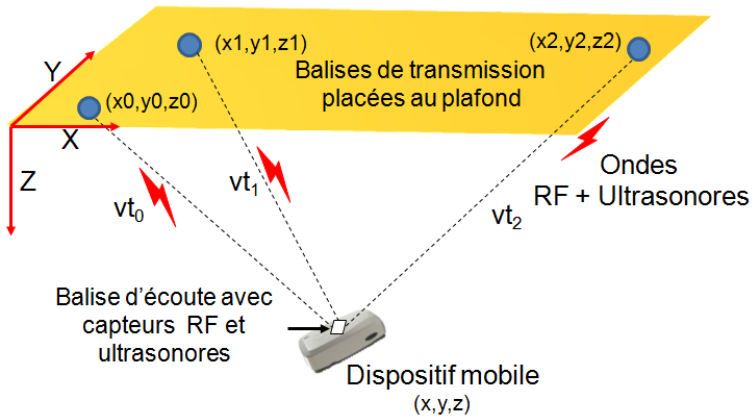

FIGURE 6.5 – Calcul de localisation par la méthode de triangulation.

6.2.3 Autres capteurs/actionneurs

Pour les besoins de nos scenarii, nous avons utilisé la gamme de produits CLEODE de la société CLEODE. Il s'agit d'une plate-forme matérielle et logicielle basée sur le réseau Zigbee.

- Prise de courant *ZPlug*

La prise de courant *ZPlug* permet de commander tout type d'appareils 220V ne dépassant pas 3500W de consommation électrique. Elle fournit également une information sur la consommation électrique de l'appareil lorsque celui-ci est activé. Cette prise de courant embarque une application permettant la commutation d'un appareil électrique par l'intermédiaire d'une commande On/Off et la mesure de la consommation de ce même appareil.

- Bracelet-montre ZCare

Un bracelet-montre émet des alertes radiofréquences sur détection de défaillance de type :

- Alerte manuelle par appui sur un bouton d'urgence ;
- Détection d'un pouls hors norme : À partir de la mesure périodique du pouls et de la valeur moyenne fixée à l'initialisation, le bracelet peut émettre une alerte ;
- Détection d'une chute : Le bracelet permet de détecter une chute à partir d'un profil d'activité fixé à l'initialisation ;

- Détecteur ZDoor

Ce détecteur permet de détecter l'ouverture/fermeture d'une porte/fenêtre.

- Système de commutation de lumière *ZLight*

Il s'agit d'un commutateur de lumière qui permet de commander deux lampes

d'une puissance maximale de 500W chacune.

- Détecteur de présence *Zmove*

Ce détecteur utilise un capteur infrarouge pour détecter des mouvements dans une pièce dans un rayon de 10 mètres maximum.

6.3 Architecture logicielle d'implémentation

L'architecture logicielle de la plateforme d'expérimentation a été développée dans le cadre du projet *SembySem* [74] [73] puis améliorée dans le cadre des projets *A2Nets* et *WoO*. Elle est composée principalement de trois blocs faiblement couplés, qui peuvent être répartis et répliqués physiquement sur plusieurs machines, figure 6.6.

1. Le bloc Façade : Ce bloc consiste en un intergiciel utilisant la technologie *Entreprise Service Bus* (ESB). Il permet de cacher d'une part, l'hétérogénéité des sources d'informations utilisant des formats et protocoles de communication différents et d'autre part, la complexité technique d'implémentation. Il permet aussi la communication avec les objets capteurs/actionneurs de l'environnement ambiant. Pour chaque événement observé dans l'environnement, un message contentant l'ensemble des données est crée puis transmis à la couche de communication. Ces données sont annotées sémantiquement à l'aide de l'ontologie *AmiOnt* décrite dans le formalisme μConcept. Les données sont ensuite transmises au noyau de raisonnement dans un flux au format standard XML. Le bloc Façade a pour fonctions :
   - La réception des données des entités du monde réel et l'envoi de notifications vers le noyau de raisonnement. Le bloc Façade permet de construire une représentation abstraite des événements observés, comme par exemple, la présence d'une personne à un endroit donné, l'appui sur un bouton d'alarme ;
   - La réception des actions envoyées par le noyau de raisonnement, comme par exemple, envoyer un message, déplacer le robot, allumer la lumière. Les descriptions sémantiques de ces actions sont ensuite transformées en commandes compréhensibles par les entités du monde réel ;
2. Le noyau de raisonnement réactif permet à partir de l'observation de connaissances contextuelles explicites et d'un ensemble de règles *SmartRules*, la déduction de connaissances implicites ou d'actions à destination des entités. Le noyau de raisonnement réactif comporte deux modules : Le module de communication et le module d'orchestration :
   - Le module de communication permet l'insertion dans le modèle sémantique des messages provenant des objets du monde réel ;
   - Le module d'orchestration permet d'ordonnancer l'exécution des modules tels que le module de vérification des contraintes d'intégrité et le moteur d'inférence Drools. Ce dernier, basé sur l'algorithme Rete, est en charge de l'exécution des règles et de la synchronisation de la base de connaissances

par rapport au modèle sémantique. Le module de vérification des contraintes d'intégrité garantit, quant à lui, la cohérence du modèle sémantique, et par conséquent, la cohérence du système ;

3. Le noyau NKRL permet d'inférer la causalité entre événements en se basant non seulement sur les observations courantes mais aussi sur les observations passées. Le lien de causalité entre contextes est établi via un mécanisme de requête-réponse. La réponse à une requête s'obtient à partir d'un ensemble de règles d'hypothèses et de transformations. Ces règles, définies dans le module *Context Query*, utilisent le principe de l'appariement (unification) entre une requête et une ou plusieurs occurrences de prédicats. L'opération d'unification est réalisée par le module *FUM* (*Filtering Unification Module*).

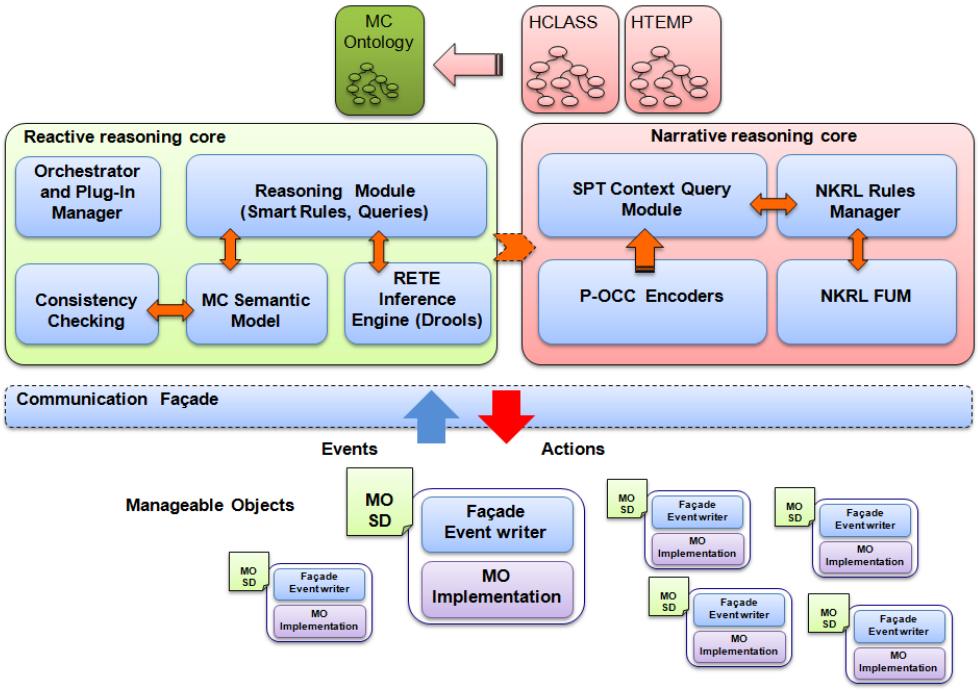

FIGURE 6.6 – Architecture globale de la plateforme.

6.4 Validation du langage μConcept selon l'hypothèse du monde fermé

Pour valider le modèle sémantique proposé dans le chapitre 4, nous décrivons tout d'abord l'ontologie *AmiOnt* pour la modélisation du contexte dans le cadre des environnements à intelligence ambiante. Cette ontologie sert également de support sémantique pour la définition de règles génériques permettant, d'une part, la reconnaissance implicite de contextes à partir de connaissances contextuelles explicites et l'adaptation au contexte.

Nous avons retenu trois cas d'utilisation dans le cadre de l'assistance à une personne âgée vivant seule :

1) Rappeler l'heure de prise de médicaments

Il s'agit ici d'un service d'assistance cognitive qui consiste à rappeler à la personne l'heure de prise des médicaments à partir de la prescription du médecin et du contexte courant. Deux informations contextuelles sont considérées ici : La localisation de la personne et sa proximité vis-à-vis d'un média. Ces deux informations contextuelles sont exploitées pour notifier à la personne la nécessité de prendre ses médicaments selon l'une des modalités suivantes : i) message vocal ou textuel via un *PC* ; ii) message vocal du robot, en déplaçant ce dernier vers l'endroit où se trouve la personne, et en supposant que cet endroit soit accessible ; l'accessibilité signifie que la porte donnant accès à cet espace est ouverte ; et iii) message SMS/MMS via le téléphone portable de la personne si cette dernière se trouve à l'extérieur de son habitation.

2) Reconnaitre une activité : Préparer un repas

Ce cas d'utilisation montre l'intérêt du langage de règles *SmarteRules* et de l'ontologie *AmiOnt* proposés pour la reconnaissance implicite du contexte et plus particulièrement les activités d'une personne, en l'occurrence ici "préparer un repas". L'adaptation à ce contexte consiste ici à prodiguer des conseils diététiques. Ces conseils sont véhiculés sous forme de messages destinés à la personne ou à un membre de sa famille.

3) Gérer une situation d'urgence

Dans ce dernier cas d'utilisation, nous supposons que la personne porte un détecteur de chute de type bracelet-montre *ZCare*. Ce dernier permet de remonter une alerte par appui sur un bouton dédié. Nous considérons ici deux situations distinctes : i) La personne chute puis active le bouton d'urgence, et ii) la personne chute mais elle n'active pas le bouton d'urgence.

Les informations contextuelles retenues dans ce cas sont :

- La localisation de la personne ;
- L'état de la personne (consciente ou inconsciente) : Cette information contextuelle est non-mesurable directement ;

  Dans la première situation, il s'agit de remonter aux personnes concernées (famille, corps médical, etc.) une alerte indiquant également la localisation de la personne dans la maison. Dans la deuxième situation, il s'agit de procéder dans un premier temps à une levée de doute, en déplaçant par exemple le robot vers l'endroit où se trouve la personne, et en tentant d'établir un dialogue avec elle. Si la personne n'interagit pas avec le robot, une alarme est alors remontée similairement à la première situation. Dans le cas contraire, le robot

peut considérer qu'il s'agit d'une fausse alerte (chute non grave) ne nécessitant pas l'intervention du corps médical ou des proches.

6.4.1 Implémentation de l'ontologie

Dans ce qui suit, nous présentons les principaux éléments de l'ontologie *AmiOnt* implémentée pour la validation du modèle décrit dans le chapitre 4. Ainsi, pour la mise en œuvre de ces scénarii, nous avons défini dans l'ontologie *AmiOnt* 68 concepts et 72 propriétés.

L'environnement intelligent ambiant est décrit à travers une représentation sémantique consistant en un ensemble de concepts décrivant les entités composant l'environnement. Nous avons montré dans le chapitre 4 que ces concepts sont organisés en taxonomie. Par exemple, les concepts *House* et *Offices_ Building* sont une spécialisation du concept *Building_ Area_ Component*. Le concept *Active_ Object* est une généralisation des concepts *Sensor, Device et Actuator*.

Relations entre concepts

Ces relations sont définies à travers des propriétés de concepts. Les figures 6.7 et 6.8 illustrent une partie des relations entre concepts définies dans l'ontologie *AmiOnt*. Ainsi, la propriété *involvedIn* définit une relation entre le concept *Person* et le concept *Human_ Activity*. La propriété *contains*, quant à elle, exprime qu'un espace donné contient des objets statiques (*Table, Chair, etc.*). Les dispositifs physiques tels qu'un détecteur de présence, un bracelet-montre Zcare, un système de localisation, sont représentés respectivement dans l'ontologie *AmiOnt* par les concepts *Presence_ Sensor, Fall_ Sensor et Localisation_ Sensor*, figure 6.8.

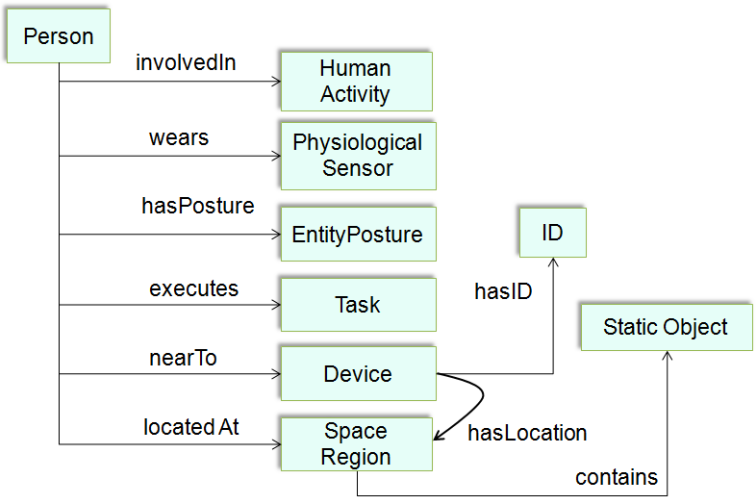

FIGURE 6.7 – Les propriétés principales du concept *Person*.

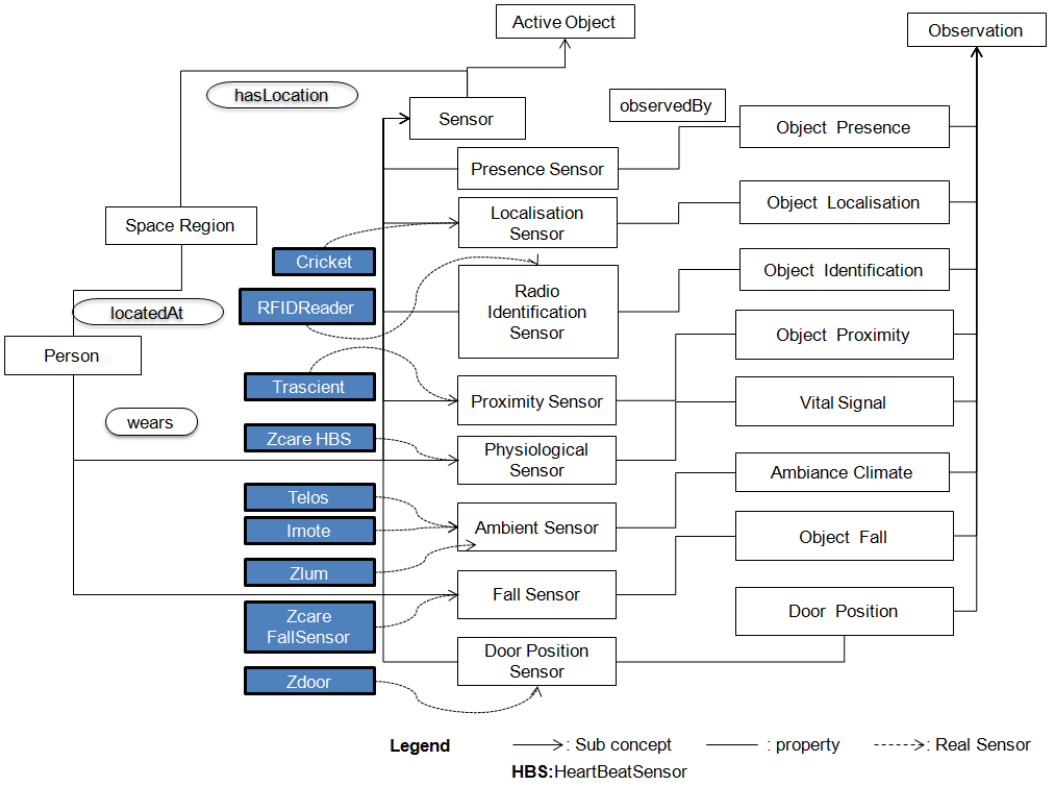

FIGURE 6.8 – Relations entre concepts.

Les instances

Les instances représentent des éléments spécifiques d'un concept. Chaque instance possède un identifiant unique qui permet de la distinguer d'une autre instance.

La figure 6.9 donne des exemples d'instances créées à partir de la plateforme ubiquitaire du laboratoire. L'instance *FALL :#1235* correspond à une instance du concept *Physiological_Sensor*. Les instances *RFID_READER :#101*, *RFID_READER#102*, correspondent à deux instances différentes du concept *Radio_Identification_Sensor*. L'instance *LIGHT :#1024* correspond, quant à elle, à une instance du concept *Ambient_Sensor*.

Pour savoir si un espace est suffisamment éclairé, la propriété *isLighted* associée au concept *Space_Region* peut être utilisée. Cette propriété est mesurable via un capteur de luminosité associé au concept *Ambient_Sensor*. À partir de la localisation de ce capteur et des relations sémantiques entre les concepts *Ambient_Sensor et Space_Region*, il est ainsi possible de déduire à un haut niveau sémantique que l'endroit *KITCHEN :#1058* qui est une instance du concept *Kitchen*, est éclairé ou non. L'espace *Kitchen* est déduit grâce à la propriété *hasLocation* associée au capteur de luminosité.

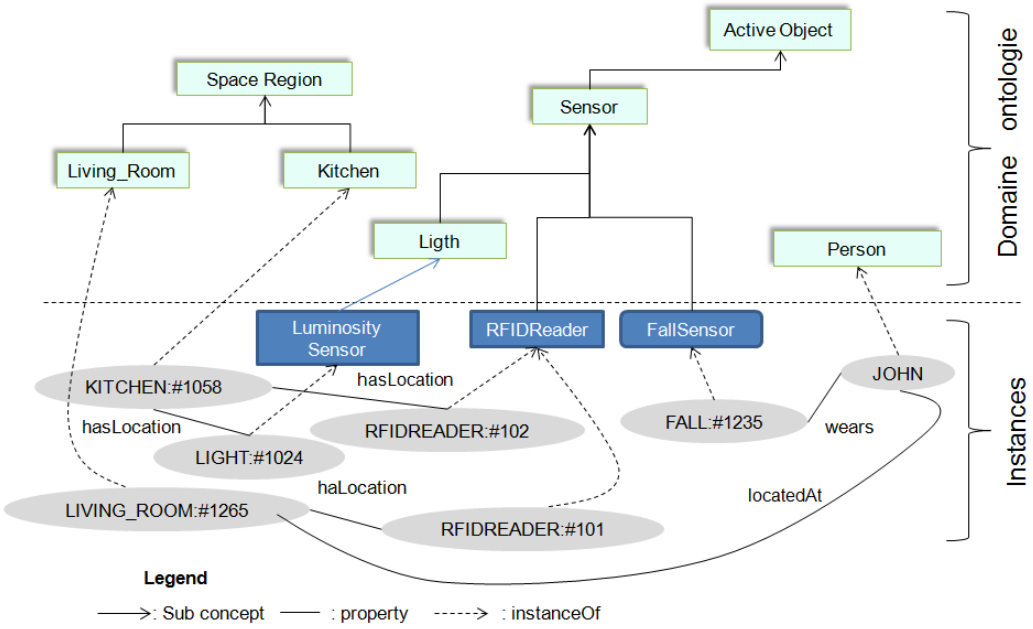

FIGURE 6.9 – Exemples d'instanciation de concepts.

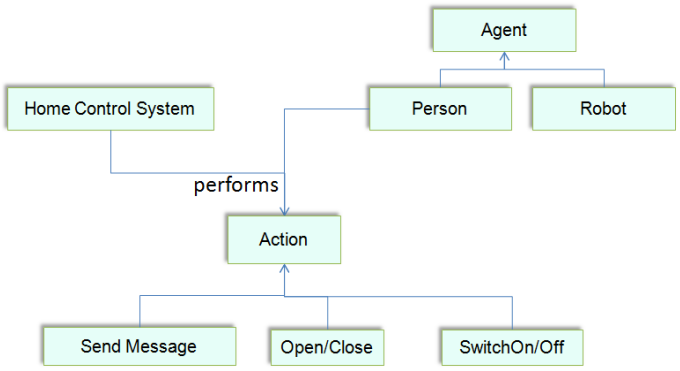

FIGURE 6.10 – Représentation des actions dans le langage μConcept.

Les actions

En plus de la représentation sémantique des objets physiques/logiques, le langage μConcept permet la représentation sémantique des actions susceptibles d'être exécutées par les entités. Dans ce modèle, une action permet d'une part, de créer, supprimer ou mettre à jour des instances dans la base de connaissances, et d'autre part, d'exprimer les interactions entre les objets eux-mêmes (éteindre la lumière, déplacer le robot, notifier un message, etc.) ou entre l'humain et les objets physiques/logiques. Par exemple, un humain peut déclencher l'ouverture d'une porte,

mettre en marche la cuisinière ou interroger le système via un écran tactile, figure 6.10.

Capteur	**Description des informations contextuelles**
Détecteur d'ouverture/fermeture de porte	Ce capteur permet de déduire qu'un espace est accessible, si la porte d'accès à cet espace est ouverte. Il peut également être utilisé pour reconnaître implicitement un contexte traduisant par exemple un problème de sécurité à partir des observations suivantes : a- La porte/fenêtre est ouverte b- La personne est à l'extérieur ; c- La personne dort ; d- Il fait nuit ;
Détecteur de présence	L'information contextuelle remontée par ce capteur permet de déduire qu'un espace est occupé ou inoccupé. Ce type d'information peut par exemple être exploitée pour déduire que la maison est vide.
Détecteur de mise sous tension d'un équipement électrique	Ce capteur consiste à déduire qu'un objet, de type équipement électrique (TV, four à microondes, cafetière, etc.), est en état de marche. À partir de cette information contextuelle, il est possible de reconnaitre un contexte de haut niveau tel que "préparer un repas".
Détecteur de chute	Ce capteur permet de déduire qu'une personne a chuté.
Système de localisation	Ce système permet d'observer les déplacements d'une personne. L'endroit où se trouve une personne peut aider le système à cerner la situation dans laquelle se trouve la personne. Par exemple, une personne ne peut pas préparer un repas si elle n'est pas dans la cuisine. De même, elle ne peut pas être sous sa douche si elle ne se trouve pas dans la salle de bain, etc.
Identification-proximité	L'information contextuelle fournie par ce capteur permet d'identifier les entités de l'environnement (objets, personne). Ce capteur permet de fournir une information contextuelle de localisation si le système de localisation ne permet pas de le faire. Ce type de connaissances permet également d'établir des relations de proximité comme par exemple la proximité d'une personne par rapport à un objet donné (téléphone, cuisinière, etc.)
Capteur de taux de luminosité	Le taux de lumière mesuré par ce capteur permet de déduire si un espace est éclairé ou non.
Capteur d'humidité	Le taux d'humidité mesuré par ce capteur permet de déduire si un espace est humide. Ce type de connaissances contextuelles peut s'avérer pertinent dans le cas par exemple d'applications de surveillance de personnes âgées allergiques aux acariens et souffrant de l'asthme.
Capteur de température	L'utilisation de l'information mesurée par ce capteur permet de déduire qu'il fait chaud/froid dans un espace donné.
Détecteur d'activation du bouton d'alarme	Permet de déduire qu'une personne est en situation d'urgence.

TABLE 6.1 – Description des informations contextuelles explicites remontées par les capteurs.

6.4.2 Utilisation du modèle μConcept pour la reconnaissance du contexte

Pour la perception et la reconnaissance du contexte, nous considérons deux types de connaissances contextuelles pouvant être manipulées :

- Les connaissances contextuelles observables, c'est-à-dire, mesurables directement. Par exemple, un capteur de température fournit une information liée au degré de température. Le modèle d'ontologie proposé permet de représenter ce type de connaissances sous forme d'instances de concepts/propriétés ;
- Les connaissances non-observables mais déductibles par inférence à partir des connaissances contextuelles observables. Par exemple, en absence d'un capteur permettant de détecter si une personne est inconsciente, un média tel que l'écran tactile du robot ou celui du téléphone, peut être utilisé pour interagir avec cette personne et déduire que cette dernière est consciente ou inconsciente ;

La représentation du contexte dans le modèle proposé s'articule autour de règles génériques. La reconnaissance d'un contexte non-observable directement est alors le résultat de l'agrégation d'un ensemble de connaissances contextuelles observables encapsulées dans des règles.

6.4.2.1 Spécification de règles pour la reconnaissance du contexte.

Dans ce qui suit, nous présentons un ensemble de règles *SmartRules* combinant des concepts et leurs relations exprimés à partir de l'ontologie *AmiOnt* présentée dans le paragraphe 6.4.1. Le tableau 6.1 décrit les dispositifs de la plateforme expérimentale utilisés et les connaissances contextuelles remontées par ces dispositifs. Ces derniers fournissent des connaissances selon divers modes : i) périodiquement dans le cas par exemple du capteur de température, ii) sur occurrence d'un événement tel que l'activation d'un bouton d'urgence ou de la détection de la présence d'une présence.

Dans ce qui suit, nous donnons quelques exemples de règles de reconnaissance du contexte. L'implémentation détaillée de ces règles est donnée en annexe B. Dans chacune de ces règles, la variable *?observer* est associée à la propriété *observedBy*. Cette dernière exprime une relation entre le concept *Observation* et un capteur, figure 6.11. Dans notre modèle, nous faisons une distinction entre les observations numériques et les observations qualitatives. Les observations numériques sont représentées par le concept *Quantitative_ Observation*. Ce dernier permet de modéliser les valeurs numériques observées par un capteur. Tandis que les observations qualitatives, sont représentées par le concept *Qualitative_ Observation*. Ce dernier représente une observation à un haut niveau sémantique. Ainsi, la valeur numérique d'une température peut être représentée par une propriété *hasValue* associée au concept *Temperature*, comme par exemple 42 degrés. La propriété *isWarm* permet, quant à elle, d'exprimer sémantiquement qu'un espace est chaud.

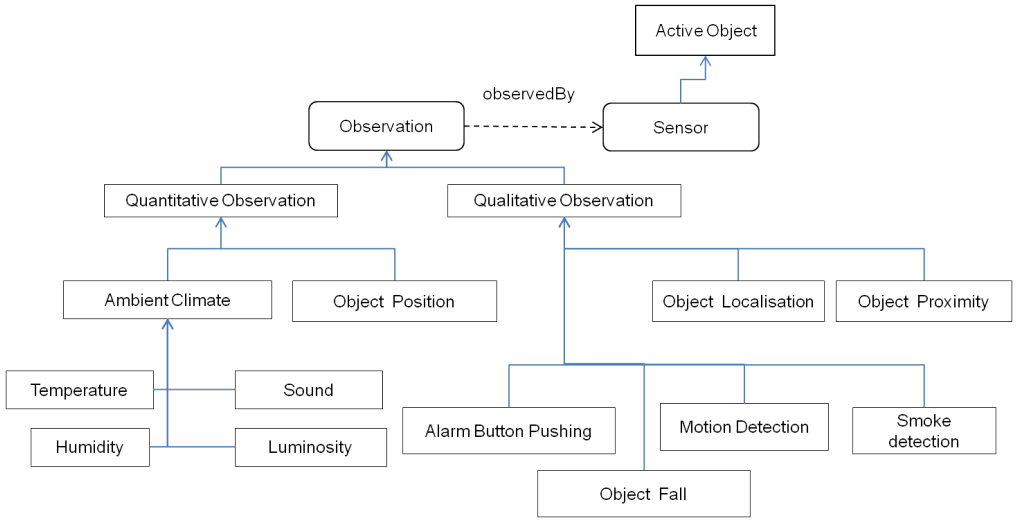

FIGURE 6.11 – Hiérarchie du concept Observation.

Détection de présence

La règle générique ci-dessous est associée au capteur de détection de présence représenté dans le modèle sémantique par le concept *Presence_Sensor*. Grâce à la propriété *hasLocation* associé au concept *Object_Presence*, la règle "Presence" permet d'inférer qu'un espace défini par la variable *?space* est occupé.

```
rule "Presence"
conditions
    Object_Presence( ?observed := observedBy) ;
    ?observed ( ?space := hasLocation) ;
actions
    ?space-> isOccupied :=true ;
    update( ?space) ;
end
```

Accessibilité d'un endroit

Il s'agit ici de déduire si un espace est accessible ou non en utilisant la règle générique suivante :

```
rule "LocationStatus"
conditions
    Door_Status ( ?observed := observedBy, not (closed)) ;
```

```
    ?observed ( ?space := hasLocation) ;
actions
    ?space->isAccessibleSpace :=true ;
    update( ?space) ;
end
```

Cette règle utilise des liens sémantiques existant entre les concepts *Space_ Region* et *Door_ Status_ Sensor*. En effet, la propriété *hasLocation* permet d'associer à un haut niveau sémantique, une localisation dans un espace donné à un capteur donné représenté dans l'ontologie *AmiOnt* par le concept *Door_ Status_ Sensor*. Cet espace est représenté à son tour par le concept *Door_ Status*. La relation à un haut niveau sémantique entre le concept *Door_ Status* et un capteur de détection d'ouverture/fermeture de porte représenté par le concept *Door_ Status_ Sensor*, est établie grâce à la propriété *observedBy*. Ainsi, la règle *LocationStatus* permet d'une part, d'associer une porte d'accès à un espace défini par la variable *?espace*, et d'autre part, d'inférer que cet espace est accessible si la porte d'accès est ouverte.

Localisation d'une personne

Cette règle générique permet de déduire que la variable *?person* associée au concept *Person* se trouve dans un espace représenté par la variable *?space*, associée au concept *Space_ Region*.

```
rule "Localisation"
conditions
    Object_Localisation( ?observer := observedBy, ?space :=hasLocation) ;
    ?person :=Person() ;
actions
    ?person->locatedAt := ?space ;
    ?person -> isAtHome := true ;
    update( ?person) ;
end
```

Proximité relative

La règle générique "ProximityObservation" est utilisée pour déduire une relation de proximité exprimée par la propriété *nearTo* du concept *Person*. Grâce à la propriété *hasID* associé au concept *Object_ Proximity*, la règle "ProximityObservation" permet d'inférer une relation de proximité entre une personne définie par la variable *?person* et un média défini ici par la variable *?deviceId*.

```
rule "ProximityObservation"
conditions
    Object_Proximity ( ?observer := observedBy) ;
```

```
    ?observer ( ?deviceId := hasId) ;
    ?person :=Person() ;
actions
    ?person -> nearTo := ?deviceId ;
    update( ?person) ;
end
```

Identification d'une entité

La règle "Identification" permet de notifier la détection d'une entité (personne, robot, objet) représentée par la variable *?entityId*. L'endroit où l'entité est détectée est représenté par la variable *?space*.

```
rule "Identification"
conditions
    Object_Identification( ?observer : = observedBy) ;
    ?observer ( ?space :=hasLocation, ?entityId :=identityOf) ;
actions
    ?objectid -> nearTo := ?space ;
    ?objectid -> hasLocation := ?space ;
end
```

Cette règle peut être spécialisée pour la détection d'une personne ou d'un robot. Pour ce faire, il suffit de lui adjoindre la condition suivante :
?objectid isInstanceOf (Robot or Person). En utilisant l'opérateur *isInstanceOf*.

Personne à l'extérieur de son habitation

La règle générique "PersonOutSide" est une forme complémentée de la règle "Localisation" utilisant l'opérateur *not exists*. Elle permet d'inférer qu'une personne définie par la variable *?p* associée au concept *Person* se trouve à l'extérieur de son habitation en positionnant la propriété *isAtHome* du concept *Person* à false.

```
rule "PersonOutside "
conditions
    not exists(Object_Localisation()) ;
    ?person :=Person() ;
actions
    ?p->isAtHome() :=false ;
    update( ?person) ;
end
```

6.4.3 Services AAL sensibles au contexte

Il s'agit ici d'exploiter les informations contextuelles issues de la perception du contexte et les règles génériques, pour fournir le(s) service(s) les plus adapté(s) à l'usager. Cette adaptation consiste à déclencher des actions à partir de ces règles génériques. Certains services sensibles au contexte nécessitent l'exécution d'une seule action, comme par exemple éteindre la lumière, envoyer un message, etc. Alors que d'autres nécessitent l'exécution d'un plan d'actions. Considérons par exemple le cas d'un aidant désirant préparer un repas à une personne en perte d'autonomie. Préparer un repas est vu ici comme un contexte non-observable directement mais pouvant être inféré en agrégeant des connaissances contextuelles observables telles que la présence d'une personne dans la cuisine, la cuisinière est en marche, etc. Par conséquent, l'adaptation à ce contexte particulier consiste à exécuter la séquence d'actions suivantes :

- Déplacer le robot vers l'endroit où se trouve la personne ;
- S'assurer que le repas est préparé pour la personne assistée. Cette information peut être déduite en interrogeant directement l'aidant ;
- Le robot peut accompagner l'aidant en lui fournissant alors des informations utiles comme le type d'aliments recommandés pour la personne assistée, etc.

Premier cas d'utilisation : Assistance cognitive du type coaching

Dans ce scénario, il s'agit d'apporter une assistance cognitive à une personne âgée en lui rappelant la nécessité de prendre ses médicaments selon la prescription du médecin.

La règle "Notification" ci-dessous décrit un service de notification utilisant le moyen de communication le plus adapté en termes de localisation courante de la personne et de proximité vis-à-vis d'un média donné (téléphone portable, écran de PC, écran du robot, synthétiseur vocal du robot, écran TV).

```
rule "Notification"
conditions
    _Alert( ?message := Message) ;
    Person( ?device := nearTo)
    ?device isInstanceOf (Device) ;
    ?device(isRunning) ;
actions
    Action ?notification :=createAction(_Notification) ;
    ?notification -> viaMedia := ?media ;
    ?notification -> message := ?message ;
    execute( ?notification) ;
end
```

En exploitant la règle de perception du contexte “ProximityObservation”, il est possible de déduire le média le plus proche de la personne via la propriété *nearTo*. La règle *Notification* crée ainsi une instance de l’action *_Notification* en spécifiant les valeurs des variables *?message et ?media* qui correspondent respectivement au message à notifier et au média sélectionné.

Second cas d’utilisation : Assistance cognitive du type coaching

Règle SmartRule	Règle CONON
rule " PrepareMeal" **conditions** ? entity := ElectricalObject (description = = “Oven”) ?entity(isRunning) Person(locatedAt = = Kitchen) **actions** Activity ?activity :=createInstance(Activity); ?activity ->involvedIn:=PrepareMeal; **end**	"PrepareMeal_2" (?u locatedIn Kitchen) ∧ (ElectricOven locatedIn Kitchen) ∧ (ElectricOven status ON) ⇒ (?u situation COOKING)

FIGURE 6.12 – Règle de reconnaissance du contexte « ok » "préparer un repas" dans les deux formalismes : SmartRules et CONON.

Dans ce scénario, nous nous intéressons à la fourniture d’un service d’assistance cognitive du type “coaching” dans le cadre d’une activité de type “préparation d’un repas”. Pour ce faire, nous nous sommes inspirés de l’ontologie CONON et de la règle SWRL “PrepareMeal_2” pour la reconnaissance de l’activité “préparer un repas”, figure 6.12.

La règle “PrepareMeal” permet de reconnaître l’activité “préparer un repas” à partir d’observations sur l’état de marche du four électrique et la présence de la personne dans la cuisine, figure 6.12. L’adaptation au contexte est décrite par la règle “HealthAdvice”. Cette dernière consiste à émettre des conseils de santé (prise de médicament, conseils diététiques) en utilisant la règle “Notification”.

```
rule "HealthAdvice"
conditions
    Human_Activity (involvedIn== PrepareMeal) ;
actions
    _Alert ?message :=createAction(_Alert) ;
    insert( ?message) ;
end
```

Troisième cas d'utilisation : Assistance cognitive en situation d'urgence

Dans ce dernier scénario, nous supposons que la personne porte un bracelet-montre *ZCare*. Ici, l'objectif consiste à procéder à une levée de doute après le déclenchement d'une alarme de type "chute d'une personne". En cas de chute, le robot se déplace vers l'endroit où la personne est localisée et tente de vérifier si cette dernière est consciente ou non. Il s'agit ici d'un contexte non-observable directement mais pouvant être déduit à partir d'un dialogue établi entre le robot et la personne. Si cette dernière n'interagit pas avec le robot, elle est considérée comme inconsciente et par conséquent, le contexte courant correspond à une situation d'urgence. L'adaptation à ce contexte consiste alors à déclencher une alarme. L'observation des contextes "Fall" et "Emergency" s'effectue à l'aide des règles génériques suivantes :

```
rule "Fall"
conditions
    ?entity := Object_ Fall() ;
    ?entity isInstanceOf(Person) ;
actions
    Fall_Event ?event :=createInstance(Fall_Event) ;
    insert( ?event) ;
end

rule "Emergency"
conditions
    RiseAlarm_Button_Pushing ( ) ;
actions
    Action ?alarm :=createInstance(_EmergencyAlarm) ;
    insert( ?alarm) ;
end

rule "CheckPersonState "
conditions
    Fall_Event() ;
    not exists (_EmergencyAlarm()) ;
actions
    Action ?personState :=createAction(_CheckPersonState) ;
    execute( ?personState) ;
end
```

Dans le cas où la personne ne déclenche pas d'alarme via le bouton d'urgence, la règle "CheckPersonState" est déclenchée pour vérifier l'état de la personne. Il s'agit ici de lever le doute en utilisant le paramètre de contexte *Space_Region*. Pour ce

faire, le système localise tout d'abord la personne à l'aide de la règle "Localisation", puis vérifie si l'espace où se trouve la personne est accessible à l'aide de la règle "AccessibleSpace". Le média de notification le plus proche est sélectionné à l'aide de la règle "ProximityObservation".

Pour évaluer la faisabilité du modèle sémantique proposé d'un point de vue réactivité, nous avons évalué le temps de réponse du système suite à une observation du contexte courant. Ce temps de réponse comprend le temps d'inférence et le temps du traitement du bloc façade (encodage des données capteurs, contrôles de contraintes d'intégrité et insertion des instances de concepts dans la base de connaissances du noyau de raisonnement, envoi d'un message à un objet actionneur). Le tableau 6.2 résume les temps obtenus pour les trois cas d'utilisation étudiés précédemment. Les performances obtenues (temps de réponse inférieurs à 3 s) sont satisfaisantes et compatibles avec la dynamique des applications visées.

Premier cas d'utilisation	Deuxième cas d'utilisation	Troisième cas d'utilisation
2.1 s	2.9 s	2.6 s

TABLE 6.2 – Temps de réponse obtenus pour les trois cas d'utilisations

6.5 Utilisation du raisonnement narratif pour la reconnaissance du contexte

Pour montrer l'intérêt du modèle narratif *NKRL* pour la reconnaissance de contextes non-observables liés au temps, nous considérons à nouveau le cas d'utilisation 3. Nous supposons ici que la personne n'a pas actionné le bouton SOS, non pas parce qu'elle est inconsciente mais parce qu'elle n'a pas entendu le message émis par le robot dû à une panne de son synthétiseur vocal. La prise de décision nécessite ici d'établir une interprétation du contexte qui n'est pas fondée uniquement sur les observations courantes mais aussi sur les observations passées. Dans la suite de ce chapitre, nous présentons les règles d'hypothèse et de transformations utilisées pour la reconnaissance implicite du contexte "Situation urgente", puis nous analysons le déroulement du processus d'inférence, tableau 6.3.

Dans ce scénario, la reconstruction du contexte "Situation urgente" se ramène à expliquer la raison pour laquelle le système a déclenché une alarme. Pour ce faire, un premier motif de recherche correspondant à la requête initiale, représentée ci-dessous, est créée à partir de la prémisse de la règle d'hypothèse.

X1) requête initiale
PREDICAT : PRODUCE
SUBJ(ect) : control_system
OBJ(ect) : triggering_
TOPIC : alarm/control_tool

Règle d'hypothèse :	**Règle de transformation 1 :**
X1) PREMISSE : PREDICATE PRODUCE SUBJ var1 OBJ triggering_ TOPIC alarm/control_tool var1 = home_control_system	**ANTECEDENT :** PREDICATE OWN SUBJ var1 OBJ control_ BENF var2 var1 = home_control_system var2 = human_being
Y1) CONDITION 1 : PREDICATE OWN SUBJ var1 OBJ control_ BENF var2 var2 = human_being	**CONSEQUENT 1 :** PREDICATE PRODUCE SUBJ var1 OBJ detection_ : (var3) TOPIC var2 var3 = building/area_component
Y2) CONDITION 2 : PREDICATE BEHAVE SUBJ var1 MODAL user_ TOPIC var3 CONTEXT (SPECIF control_ var2) var3 = robot_	**CONSEQUENT 2 :** PREDICATE OWN SUBJ var3 OBJ property_ TOPIC (SPECIF part_of (SPECIF var4 var2)) var4 = house_
Y3) CONDITION 3 : PREDICATE PRODUCE SUBJ var2 OBJ button_pushing TOPIC var4 { oblig } var4 = emergency_button	**Règle de transformation 2 :** **ANTECEDENT :** PREDICATE BEHAVE SUBJ var1 MODAL user_ TOPIC var2 CONTEXT (SPECIF control_ var3) var1 =home_ control_system var2 = robot_ var3 = human_being
Y4) CONDITION 4 : PREDICATE OWN SUBJ var4 OBJ property_ TOPIC (SPECIF part_of (SPECIF var5 var3)) var5 = alarm/control_tool	**CONSEQUENT 1 :** PREDICATE MOVE SUBJ var3 OBJ information_content BENF var1 MODAL (SPECIF var4 var2) var4 = alarm/control_tool
Y5) CONDITION 5 : PREDICATE PRODUCE SUBJ var2 OBJ button_pushing TOPIC var4 { negv }	

TABLE 6.3 – Règle d'hypothèse et règles de transformation utilisées pour lever le doute sur une situation d'urgence

Dans cette requête X1, le prédicat $PRODUCE$ est utilisé pour déterminer l'initiateur de l'événement correspondant au déclenchement de l'alarme. Le rôle $SUBJ(ect)$ représente l'initiateur de cet événement. Le concept $control\_system$ est utilisé ici comme attribut dans le rôle $SUBJ(ect)$. Le concept $triggering\_$, attribut du rôle $OBJ(ect)$ représente le type d'action, en l'occurrence, déclencher une alarme. Le concept $alarm/control\_tool$ du rôle $TOPIC$ explicite que cet événement correspond à une alarme. La réponse à la requête initiale est comme suit, figure 6.13 :

FIGURE 6.13 – Occurrence du prédicat $PRODUCE$ précisant l'initiateur du déclenchement de l'alarme.

L'occurrence aal10.c30 du prédicat $PRODUCE$ indique que le système représenté par le symbole $HOME\_CONTROL\_SYSTEM\_1$ est à l'origine (initiateur) de l'alarme. Cette connaissance est représentée à l'aide du rôle $SUBJ(ect)$. Le symbole $ALARM\_SITUATION\_1$, attribut du rôle $CONTEXT$ permet, quant à lui, d'apporter des précisions sur la nature de l'alarme.

En s'appuyant sur l'ontologie $HClass$ et en utilisant la contrainte imposée à la variable $var1$ dans la prémisse de la règle d'hypothèse, tableau 6.3, le symbole $HOME\_CONTROL\_SYSTEM\_1$ est validé par le module de contrôle de consistance $NKRL$ étant donné que ce symbole est une instance du concept $home\_control\_system$.

De plus, le module de contrôle de contraintes, d'une part, vérifie que le symbole $ALARM\_SITUATION\_1$ est une instance du concept $alarm/control\_tool$, et d'autre part, établit une hiérarchie de concepts entre le concept $emergency\_alarm$ et le concept $alarm/control\_tool$. Cette vérification s'effectue à partir de la relation de généralisation/hiérarchisation entre concepts/instances de l'ontologie $HClass$,

formellement, *emergency_alarm* $\sqsubset$ *alarm/control_tool*.

L'occurrence aal10.c30 du prédicat $PRODUCE$ est alors retenue comme réponse à la requête initiale X1. Cette réponse clos le traitement de la partie prémisse de la règle d'hypothèse et permet de poursuivre le traitement de la condition 1 de la règle d'hypothèse.

Traitement de la condition 1 de la règle d'hypothèse

Dans ce pas de raisonnement, il s'agit d'inférer que le système représenté par $HOME\_CONTROL\_SYSTEM\_1$ est dédié à l'assistance et à la surveillance de l'habitat d'une personne. Pour ce faire, à partir de l'occurrence de prédicat aal10.c30, l'instance $HOME\_CONTROL\_SYSTEM\_1$ est assignée à la variable *var1* définie dans la condition 1 de la règle d'hypothèse. Le concept *human_being*, quant à lui, est assigné à la variable *var*2 du rôle $BENF(iciary)$. Ce dernier représente le bénéficiaire de l'assistance du système *HOME_CONTROL_SYSTEM_1.* Par conséquent, le motif de recherche Y1, associé à la condition 1 de la règle d'hypothèse, s'écrit :

PREDICAT : OWN
SUBJ(ect) : HOME_CONTROL_SYSTEM_1
OBJ(ect) : control_
BENF : human_being

Comme il n'existe dans la base de connaissances aucune occurrence du prédicat OWN satisfaisant ce pas de raisonnement (condition 1), il est nécessaire de recourir aux règles de transformation. La recherche d'occurrences du prédicat OWN satisfaisant le motif de recherche Y1 à partir de la règle de transformation 1, tableau 6.3, permet d'obtenir la réponse suivante, figure 6.14 :

L'occurrence aal8.c59 du prédicat $PRODUCE$, figure 6.14 est validée étant donné que :

1- La propriété *detection_* définie dans le rôle $OBJ(ect)$ de l'occurrence aal8.c59 du prédicat $PRODUCE$ est appariée avec la propriété *detection_* définie dans le rôle $OBJ(ect)$ figurant dans le conséquent 1 de la règle de transformation 1.

2- Le symbole $LIVING\_ROOM\_1$ est une instance du concept *building/area_component* du fait de la contrainte imposée à la variable *var3* définie dans le conséquent 1 de la règle de transformation 1. Notons ici, que le concept *building/area_component* est équivalent au concept *_Building_Area_Component* défini dans l'ontologie *AmiOnt*.

3- Le symbole $JOHN\_$ est une instance du concept *human_being*.

L'occurrence aal8.c59 du prédicat $PRODUCE$ permet alors de déduire la présence de $JOHN\_$ dans le séjour. Cette connaissance est représentée par le rôle

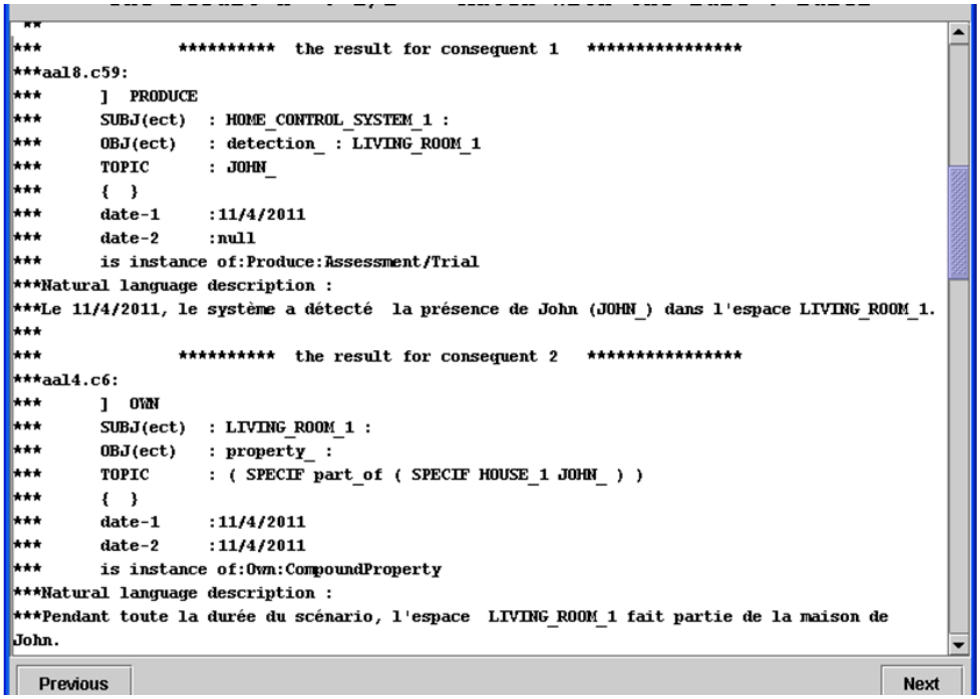

FIGURE 6.14 – Réponses aux requêtes associées aux conséquent 1 et conséquent 2 de la règle de transformation 1.

$OBJ(ject)$ ayant comme propriété le concept *detection_* .

Pour inférer que le système représenté par le symbole *HOME_CONTROL_SYSTEM_1* est dédié à l'assistance et à la surveillance de l'habitation de $JOHN\_$, le second conséquent de la règle de transformation 1 est utilisé pour vérifier que l'espace où la présence de $JOHN\_$ est détectée fait partie de l'habitation de ce dernier. L'occurrence aal4.c6 du prédicat OWN représente le résultat du déclenchement du conséquent 2 de la règle de transformation, tableau 6.3. Ce résultat permet de conclure que l'espace représenté par le symbole $LIVING\_ROOM\_1$ se trouve dans l'habitat de $JOHN\_$. À partir de l'ontologie *HClass* et de la propriété *part_of*, le module de contrôle de contraintes vérifie, d'une part, que l'espace $LIVING\_ROOM\_1$ fait partie de l'instance *HOUSE_1* et d'autre part, que l'instance *HOUSE_1* est une instance du concept *house_*. Cette vérification s'obtient en utilisant la relation de généralisation/hiérarchisation entre concepts/instances de l'ontologie *HClass*, formellement, *LIVING_ROOM_1* $\sqsubset$ $HOUSE\_1$ $\sqsubset$*building/area_component* $\sqsubset$ *house_*. Après transformation de la requête correspondant au motif de recherche Y1 de la règle d'hypothèse, la procédure d'inférence aboutit à une réponse permettant ainsi de poursuivre le traitement de la condition 2 de la règle d'hypothèse.

Traitement de la condition 2 de la règle d'hypothèse

Ce pas de raisonnement a pour objectif d'inférer que le système représenté par HOME_CONTROL_SYSTEM_1 a tenté de communiquer avec la personne. Pour ce faire, le motif de recherche Y2 ci-dessous est construit :

PREDICAT BEHAVE
SUBJ(ect) : HOME_CONTROL_SYSTEM_1 :
MODAL(ity) : user_
TOPIC : robot_
CONTEXT : (SPECIF control_ JOHN_)

Comme il n'existe aucune occurrence du prédicat $BEHAVE$ satisfaisant la requête correspondant au motif de recherche Y2, la réponse à cette requête s'obtient alors au moyen des règles de transformation. La recherche d'occurrences du prédicat $BEHAVE$ satisfaisant ce motif de recherche se ramène alors à explorer l'historique des observations pour déterminer si le système $HOME\_CONTROL\_SYSTEM\_1$ a utilisé un robot comme média pour communiquer avec $JOHN\_$. Le résultat de cette inférence consiste en une seule occurrence ayant pour identifiant aal7.c23, figure 6.15.

L'occurrence aal7.c23 du prédicat $BEHAVE$ indique que le système informe la personne $JOHN\_$ qu'il peut utiliser l'écran tactile du robot pour communiquer un message au système $HOME\_CONTROL\_SYSTEM\_1$. Cette déduction est établie comme suit :

1- Le prédicat $MOVE$ est utilisé pour indiquer la possibilité d'envoyer un message.

2- Le rôle $SUBJ(ject)$ représente l'initiateur de l'événement correspondant à la transmission du message.

3- Le rôle $BENF$ indique ici le destinataire du message, en l'occurrence, $HOME\_CONTROL\_SYSTEM\_1$.

Enfin, le concept $touch\_screen$ indique la modalité de transmission du message. Ainsi, la connaissance indiquant que la personne $JOHN\_$ peut utiliser l'écran tactile du robot est obtenue comme suit :

Le module de contrôle de contraintes vérifie que le symbole $ROBOT\_KOMPAI$ est une instance du concept $robot\_$, et que le concept $touch\_screen$ est un sous-concept de $alarm/cotrol\_tool$, en utilisant la relation de généralisation/hiérarchisation entre concepts/instances de l'ontologie $HClass$.

Ce résultat clos le traitement de la partie condition 2 de la règle d'hypothèse et permet de poursuivre le traitement des conditions 3 et 4 de la règle d'hypothèse.

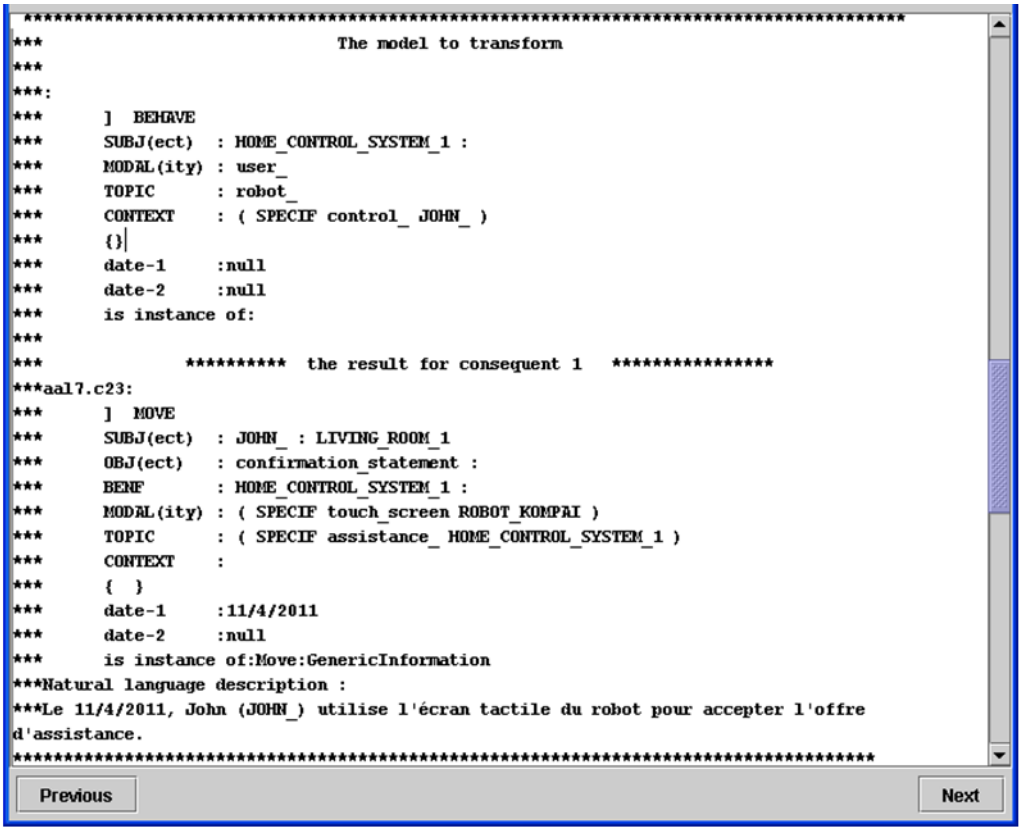

FIGURE 6.15 – Réponse obtenue à la requête associée au conséquent 1 de la règle de transformation 2.

Traitement des conditions 3 et 4 de la règle d'hypothèse

Ces deux pas de raisonnement visent à apporter des précisions sur le contenu du message émis par le robot à l'attention de $JOHN\_$. Pour ce faire, le motif de recherche Y3 est crée à partir de la condition 3 de la règle d'hypothèse :

PREDICAT PRODUCE
SUBJ(ect) : JOHN_
OBJ(ect) : button_pushing
TOPIC : emergency_alarm

Ce motif de recherche Y3 permet d'explorer l'historique des observations pour trouver des occurrences du prédicat $PRODUCE$ où $JOHN\_$ doit utiliser un bouton d'urgence. Le traitement de ce motif conduit à la réponse donnée figure 6.16.

L'occurrence aal10.c25 du prédicat $PRODUCE$ permet alors d'inférer que $JOHN\_$ doit utiliser le bouton représenté par le symbole $LIFE\_SAVING\_BUTTON\_1$. Le module de contrôle valide cette occurrence

étant donné que le symbole *LIFE_SAVING_BUTTON_1* est une instance du concept *button_pushing*. L'occurrence aal10.c25 du prédicat *PRODUCE* est alors retenue comme réponse au motif de recherche Y3 correspondant, et le traitement de la condition 4 de la règle d'hypothèse est ainsi poursuivi. Cette dernière permet de vérifier si le bouton *LIFE_SAVING_BUTTON_1* se situe sur l'écran tactile du robot. Pour ce faire, le prédicat *OWN* est utilisé car il permet de représenter la relation de possession entre l'instance *LIFE_SAVING_BUTTON_1* et le concept *touch_screen*. Par conséquent, le motif de recherche Y4, associé à la condition 4 de la règle d'hypothèse, s'écrit :

PREDICAT OWN
SUBJ(ect) : LIFE_SAVING_BUTTON_1
OBJ(ect) : property_
TOPIC :(SPECIF part_of (SPECIF alarm/control_tool
ROBOT_KOMPAI)

La réponse à la requête associée à la condition 4 de la règle d'hypothèse est comme suit, figure 6.16 :

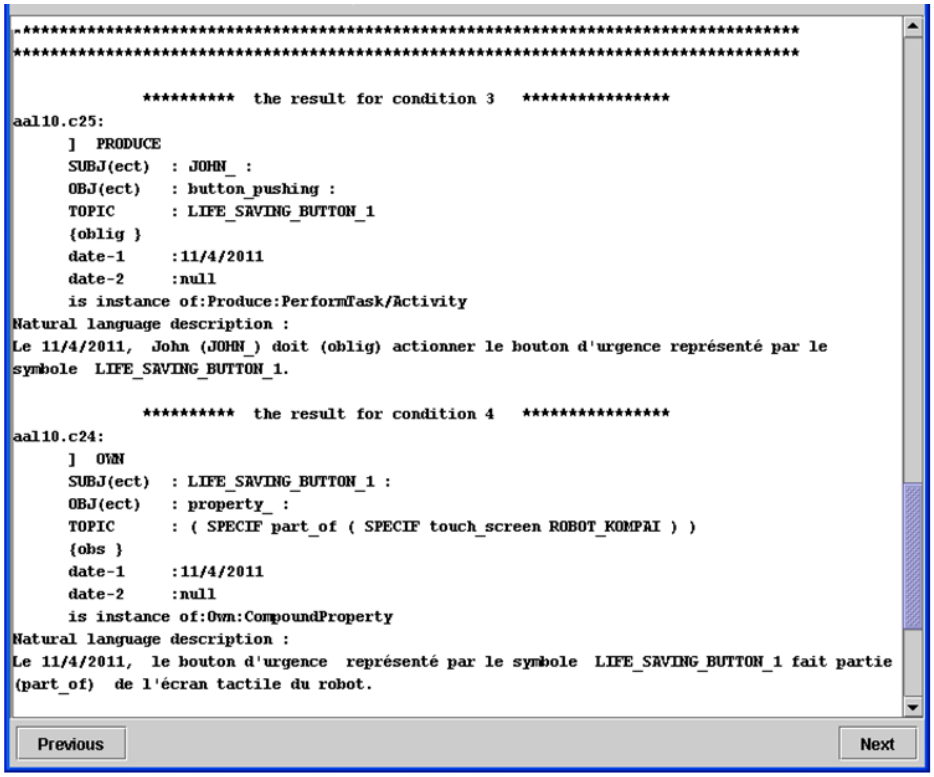

FIGURE 6.16 – Réponses aux requêtes correspondant aux condition 3 et condition 4 de la règle d'hypothèse.

L'occurrence aal10.c24 du prédicat *OWN* indique que le bouton *LIFE_SAVING_BUTTON_1* est situé sur l'écran tactile du *ROBOT_KOMPAI*. Cette déduction est établie grâce à la propriété part_of. Cette dernière permet au module de vérification de contraintes de vérifier que le bouton *LIFE_SAVING_BUTTON_1* fait partie de l'écran tactile du robot. Notons ici l'utilisation du modulateur *oblig* pour exprimer la nécessité d'actionner le bouton d'urgence si une personne se trouve en situation d'urgence.

Avant de conclure que *JOHN_* se trouve dans une situation d'urgence, le moteur d'inférence doit d'abord vérifier, après traitement de la condition 5 de la règle d'hypothèse, que *JOHN_* n'a pas actionné le bouton d'urgence.

Traitement de la condition 5 de la règle d'hypothèse

Dans le scénario 3, une situation d'urgence peut être déduite si la personne n'interagit pas avec le robot. Par conséquent, le moteur d'inférence doit explorer l'historique des observations pour trouver des occurrences du prédicat *PRODUCE* prouvant que *JOHN_* n'a pas actionné le bouton d'urgence. Pour ce faire, le motif de recherche Y5 ci-dessous, est construit à partir de la condition 5 de la règle d'hypothèse :

PREDICAT PRODUCE
SUBJ(ect) : JOHN_
OBJ(ect) : button_pushing
TOPIC : LIFE_SAVING_BUTTON_1
{ negv }

Le traitement de ce motif conduit la réponse suivante :

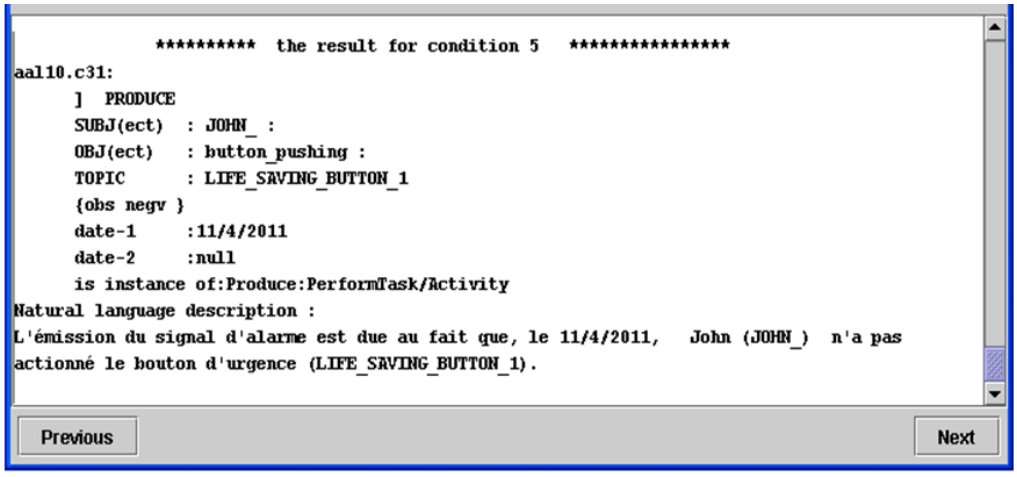

FIGURE 6.17 – Réponse à la requête associée à la condition 5 de la règle d'hypothèse.

L'occurrence aal10.c31 du prédicat *PRODUCE* indique que *JOHN_* n'a pas actionné le bouton d'urgence, figure 6.17. Ceci s'obtient grâce au modulateur *negv* exprimant la négation d'un événement. Le traitement de la règle d'hypothèse à partir de la requête initiale X1 conduit à la reconnaissance du contexte *Situtaion d'urgence*.

Remarques :

- Contrairement au langage OWL, l'utilisation de la notion de variable permet de généraliser l'application des règles de transformations et d'hypothèses dans d'autres applications d'intelligence ambiante [117]. Par exemple, il suffit d'ajouter le concept *human_being* à la variable *var1* de la prémisse de la règle d'hypothèse utilisée pour la levée de doute pour déduire si l'initiateur de l'alarme est un humain. De même, il suffit d'ajouter les concepts *offices_building* et/ou *factory_building* à la variable *var 2* de la règle de transformation pour la levée de doute, pour que cette règle puisse s'appliquer dans d'autres applications impliquant d'autres populations d'usagers et/ou d'autres lieux sociaux ou commerciaux.

- Concernant les performances temporelles du modèle NKRL, nous avons utilisé la totalité des concepts des ontologies HClass et HTemp. Pour rappel, l'ontologie HClass contient 2700 concepts et l'ontologie Htemp 150 templates. Le temps de réponse nécessaire à la reconnaissance du contexte dans le troisième cas d'utilisation (gestion de situation d'urgence et levée de doute) est de 3.7 s. Ce temps est acceptable dans le cadre de l'application étudiée. La reconnaissance de certains contextes liés par exemple à l'état physique ou mental de personnes dépendantes peut nécessiter des observations à moyen ou long terme et par conséquent, elle n'implique pas de contraintes fortes en termes de temps de raisonnement.

6.6 Conclusion

Dans ce chapitre, nous avons présenté la mise en œuvre des modèles proposés dans les chapitres 4 et 5 pour la gestion du contexte, avec l'objectif de montrer leurs intérêts et d'évaluer leurs faisabilités.

Concernant le modèle sémantique proposé dans le chapitre 4, nous avons ainsi décrit l'implémentation de l'ontologie *AmiOnt*. À travers les différents scénarii étudiés, nous avons montré que ce modèle offre la possibilité d'établir des règles génériques permettant d'une part, d'inférer des contextes non-observables à partir de connaissances contextuelles observables et d'autre part, de déclencher des actions d'adaptation au contexte. La spécialisation des règles pour inférer des contextes particuliers est relativement simple et nécessite seulement de remplacer, dans les règles *SmartRules*, les concepts *Observation* et *Action* par des sous-concepts plus

spécifiques. Grâce à l'hypothèse du nom unique, une instance de ces sous-concepts possède une représentation dans l'ontologie qui permet d'identifier de manière unique le capteur ou l'actionneur correspondant dans le monde réel. Par ailleurs, nous avons montré que l'application du principe de la négation par l'échec (NAF), qui est une conséquence de l'utilisation du raisonnement non-monotone, permet de gérer l'absence d'instances dans la base de connaissances, et ainsi, de garantir la cohérence du système.

Enfin, nous avons montré à travers le scénario de gestion d'une situation d'urgence par levée de doute que le système de raisonnement narratif permet la représentation sémantique de contextes complexes à l'aide d'un minimum d'annotations sémantiques. Nous avons montré comment des règles d'hypothèses peuvent être utilisées pour déduire des liens sémantiques entre événements et inférer qu'un contexte telle qu'une situation d'urgence est plausible. Associer des règles de transformations à des règles d'hypothèses permet d'exploiter le haut niveau d'expressivité sémantique de NKRL pour déduire de manière plausible un contexte.

Chapitre 7

Conclusion générale et perspectives

Les travaux de recherche présentés dans ce mémoire ont fait l'objet de deux contributions majeures. La première concerne la proposition d'une approche de modélisation sémantique et de raisonnement réactif fondée sur l'hypothèse du monde fermé avec supposition du nom unique pour la gestion du contexte. L'utilisation du langage μConcept permet de répondre aux limites d'utilisation du langage OWL dans les applications d'intelligence ambiante et de robotique ubiquitaire. En effet, ce type d'applications nécessite d'une part, un raisonnement réactif non-monotone et d'autre part, une représentation d'instances de concepts et d'actions selon la supposition du nom unique. Le modèle μConcept a été complété par l'ontologie *AmiOnt* qui est composée de concepts de sens commun permettant de décrire de façon générique et expressive toutes les entités peuplant un environnement intelligent ambiant, et toute information, action ou événement pouvant être observée à partir de capteurs logiques ou physiques. Pour ce faire, nous avons réutilisé des concepts proposés dans les ontologies W3C SSN, DOLCE Ultra Lite (DUL) et NKRL HClass. L'ontologie *AmiOnt* est utilisée comme support pour définir des règles de sensibilité au contexte, représentées en langage *SmartRules*. La gestion sémantique du contexte reposent sur deux types de règles : (i)les règles génériques de reconnaissance du contexte à partir d'observations courantes, et (ii) les règles génériques d'adaptation au contexte. Le modèle sémantique proposé permet de mettre à jour ou d'étendre facilement des modèles de contextes par simple modification, suppression, spécialisation ou ajout de règles *SmartRules*. D'un point de vue standardisation, le modèle proposé est construit au dessus du langage RDF-S. Il est par conséquent sémantiquement et syntaxiquement compatible avec la norme RDF-S. Enfin, le modèle proposé offre un niveau d'expressivité élevé puisque d'une part, il permet d'associer aux connaissances contextuelles une sémantique formelle claire, et d'autre part, de prendre en compte de nouveaux types de propriétés, telle que les restrictions de cardinalité, la possibilité de définir des valeurs par défaut, etc. La mise en œuvre de scénarii d'assistance cognitive nous a permis de montrer la validité du modèle proposé.

La deuxième contribution complète la première avec la proposition du modèle narratif NKRL pour la gestion du contexte. Ce modèle se distingue des modèles ontologiques existants par le fait qu'il intègre l'ontologie HTemp. Cette dernière utilise des structures hiérarchiques de prédicats et de rôles sémantiques pour la

représentation des événements dynamiques. NKRL offre un moyen de reconstituer le contexte à partir de l'historique des connaissances contextuelles. Il offre en plus des mécanismes d'inférence permettant d'établir des relations sémantiques implicites ou explicites entre ces connaissances contextuelles. Le scénario de gestion d'une situation d'urgence par levée de doute que nous avons mis en œuvre a permis de montrer le haut niveau d'expressivité du modèle NKRL ; ce dernier nécessitant peu d'annotations sémantiques pour la représentation de contextes complexes. Par ailleurs, l'association de règles de transformations aux règles d'hypothèses permet d'aboutir à des raisonnements avancés pour déduire de manière plausible un contexte. Enfin, contrairement au langage OWL, l'utilisation de la notion de variable dans NKRL permet de généraliser l'application des règles de transformations et d'hypothèses dans d'autres types d'applications d'intelligence ambiante impliquant d'autres populations d'usagers et/ou d'autres lieux sociaux ou commerciaux.

Sur la base du travail réalisé, nous pouvons dresser plusieurs perspectives de recherche à court, à moyen et à long terme. Les perspectives à court terme découlant de ces travaux de thèse concernent principalement l'extension et la mise en œuvre de l'ontologie *AmiOnt* et des règles *SmartRules* pour valider l'approche de modélisation sémantique et de raisonnement réactif dans d'autres applications sensibles au contexte. Il s'agit ici de considérer d'une part, d'autres populations d'usagers comme par exemple les personnes handicapées ou les enfants, et d'autre part, d'autres lieux sociaux tels que les maisons de retraite, les espaces commerciaux, les centres de soins, les espaces urbains, etc.

Les perspectives à moyen terme concernent tout d'abord la planification de tâches qui est une fonction essentielle pour tout système devant évoluer de manière autonome dans un environnement dynamique. Plusieurs approches ont été proposées dans la littérature pour utiliser des ontologies web sémantique dans les systèmes de planification. Bien que les ontologies permettent d'augmenter le degré d'expressivité des langages de planification, elles demeurent néanmoins incompatibles avec l'hypothèse de raisonnement en monde fermé telle qu'utilisée par les systèmes de planification. Nous proposons donc d'étendre le langage de règles *SmartRules* pour composer des plans d'assistance sensibles au contexte sous la forme de règles de type *Event-Condition-Action* ; ces plans pouvant être exécutés directement par le module de raisonnement réactif. Nous proposons d'étudier aussi la faisabilité de s'interfacer avec les plateformes de planification utilisées en robotique autonome, telles que CRAM (Cognitive Robot Abstract Machine). Il s'agit ici de déléguer l'exécution de certaines tâches au système de planification du robot ou du système intelligent ambiant. Du point de vue interaction homme-robot, il nous semble pertinent d'aller vers des modes d'interaction plus naturels en exploitant la grande expressivité et le raisonnement narratif du modèle NKRL. Il s'agit ici de développer un module de raisonnement sensible au contexte qui soit capable de convertir automatiquement un contenu exprimé en langage naturel vers la représentation NKRL et inversement, de cette dernière, vers le langage naturel.

Ce travail fait actuellement l'objet d'une thèse menée au laboratoire.
Nous pensons aussi que la modélisation des relations spatiales de l'algèbre RCC-8 sous forme de concepts de l'ontologie HClass permettra de modéliser des templates de règles pour le raisonnement spatial. Ces templates visent à simplifier l'établissement des règles de reconnaissance de contextes, et par conséquent, à réduire la complexité du processus de raisonnement dans des applications impliquant des interactions multiple entre plusieurs acteurs (humains, robots et environnement).

Concernant les perspectives à long terme, nous pensons qu'il serait intéressant d'intégrer formellement de nouveaux types de prédicats pour modéliser d'une part, la notion abstraite du futur et d'autre part, des évènements qui continuent dans le temps. L'objectif ici est d'étendre le modèle NKRL du point de vue raisonnement, pour effectuer, de manière similaire au raisonnement *Event Calculus*, des projections dans le temps et des inférences sur les effets des actions en fonction du contexte déduit. Enfin, il nous semble intéressant d'effectuer un travail de formalisation du langage NKRL, pour définir un format canonique permettant l'interopérabilité des connaissances exprimées à l'aide de ce langage avec d'autres systèmes de raisonnement basés sur la logique du premier ordre. Par ailleurs, l'utilisation de fragments de la logique du premier ordre dans la formalisation de NKRL peut être utile pour prouver la décidabilité d'un raisonnement NKRL dans le cadre d'applications critiques.

7.1 Liste des contributions

Sabri, L., Chibani, A., Amirat, Y. and Zarri, G. P., Narrative reasoning for cognitive ubiquitous robots, in : Knowledge Representation for Autonomous Robots, Workshop at IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2011), San Fransisco, USA, 2011.

Sabri, L., Chibani, A., Amirat, Y. and Zarri, G. P., Semantic and architectural approach for Spatio-Temporal Reasoning in Ambient Assisted Living, in : International Joint Conference on Artificial Intelligence (IJCAI 2011), Barcelona, Spain, pages 77-84, 2011.

Sabri, L., Chibani, A., Amirat, Y. and Zarri, G. P., Semantic reasoning framework to supervise and manage contexts and objects in pervasive computing environments, in : Proceedings of the International Conference on Advanced Information Networking and Applications (AINA 2010), Biopolis, Singapor, pages 47-52, 2011.

Sabri, L., Chibani, A., Amirat, Y. and Zarri, G. P., Semantic-based situation recognition for ubiquitous robots and ambient intelligence environments, in : Journées Nationales de la Robotique Interactive (JNRI 2011) du GDR Robotique, Paris, France, 21-22 novembre 2011.,

Sabri, L., Chibani, A., Patkos, T., Sebbak, Faouzi and Amirat, Y., Advanced Context modeling and Reasoning for ambient intelligence environments and ubiquitous robotics., in : Journées Nationales de la Robotique Interactive (JNRI 2011) du GDR Robotique, Paris, France, 2011.

Zarri, G. P., Sabri, L. and Chibani, A., Semantic-Based Industrial Engineering : Problems and Solutions, in : Proceedings of the 4th Conference on Complex, Intelligent and Software Intensive Systems (CISIS 2010), Cracovie, Poland, 2010.

Sabri, L., Chibani, A., Beck, J., Zarri, G. P., Brunner, J. S. and Gatellier, P., An Editor for Micro-Concept Rules Design, in : Proceedings of the 3rd International RuleML-2009 Challenge, collocated with the 3rd International Symposium on Rules, Applications and Interoperability (RuleML-2009), Las Vegas, USA, 2009

Annexe A

Architecture du bloc Façade

A.1 Bloc façade

Le bloc Façade [1] a pour objectif de cacher l'hétérogénéité des objets capteurs et actionneurs et assurer un canal de communication bidirectionnelle avec ces objets.

Dans l'architecture logicielle présentée dans le chapitre 5, le bloc façade a pour rôle de gérer tous les aspects concernant la communication entre le noyau de raisonnement et les objets du monde réel. Il garantit ainsi la synchronisation entre l'état des objets et leurs représentations conceptuelles dans le modèle sémantique. Le bloc façade assure aussi la transmission et l'interprétation des actions (envoyer un message, déplacer le robot, allumer la lumière, etc.) à destination des objets actionneurs du monde réel. Il permet également d'abstraire les technologies, les protocoles de communication, la découverte et l'enregistrement dynamiques de nouveaux objets ou services. Il permet aussi l'acquisition des données brutes émises par des objets ou des services. Ces données sont encodées dans un format compréhensible par les modules de raisonnement réactif et narratif.

Actuellement, le support de communication mis en ouvre au niveau du bloc façade utilise trois protocoles : *JMS, HTTP et SOAP*. Cependant, l'extension de ce bloc pour supporter d'autres protocoles tels qu'*OSGi, iPOJO, UPnP, SNMP* reste possible.

Dans le bloc façade, nous dénombrons plusieurs composants, figure A.1 :

1. Source d'événements (*Events sources*) : Ce composant reçoit des événements déclenchés par un objet du monde réel et alimente le *Contrôleur d'événements* (*Event Controller*) avec un nouveau message ;
2. Encodeur d'événements (*Event Encoder*) : Cet encodeur gère les transformations des messages en format μConcept ;
3. Contrôleur d'événements (*Event Controller*) : Ce composant traite les messages encodés par le composant *Event Controller* et invoque le composant *MsgSender* pour rediriger les messages vers le noyau ;
4. Récepteur de message (*μConcept Message Receiver*) : Ce composant a pour rôle de réceptionner des messages provenant du noyau ;
5. Gestionnaire d'actions (*Action Handler*) : Ce composant transforme chaque action contenue dans un message en commandes à exécuter par les objets du monde réel, puis invoque le composant *Action Destinations* ;

1. Le terme Façade est inspiré du domaine de la conception logicielle (design patterns)

6. Destination d'Action (*Action Destinations*) : Ce composant a pour rôle d'identifier l'objet auquel le message est destiné, et le protocole de communication utilisé ;

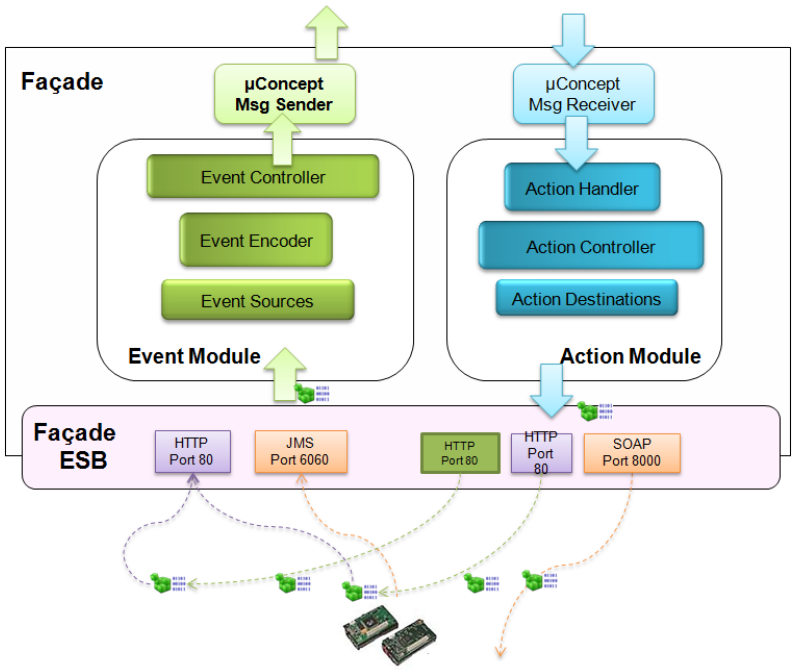

FIGURE A.1 – Architecture globale de la couche Façade.

A.1.1 Noyau de raisonnement réactif

Il offre un ensemble d'interfaces pour alimenter et interroger la base de connaissances. La principale caractéristique de ce noyau est qu'il s'appuie sur un mécanisme de plug-in. Par ailleurs, il peut être exécuté indépendamment des autres couches permettant ainsi à tous les autres composants d'utiliser une interface de communication synchrone pour échanger des connaissances via le protocole JMS. L'architecture interne de ce noyau est composée des modules suivants :

a) Modèle sémantique

Pour garantir la cohérence du modèle et du système, le modèle sémantique est associé à un module de gestion de contraintes d'intégrité (Constraint Checking), basé sur l'hypothèse du monde fermé avec présupposition du nom unique. En effet, à la réception d'un message de la façade ou du moteur d'inférence, le module vérifie si les contraintes d'intégrité (cardinalités de restrictions, domaines de propriété, propriété fonctionnelles, etc.) sont satisfaites avant de modifier le modèle, figure A.2. Ainsi, toutes les opérations de modification du modèle sémantique sont contrôlées. La vérification de contraintes d'intégrité est associée à un mécanisme de transaction qui

permet d'annuler des modifications du modèle en cas d'incohérence ou de détection d'erreur.

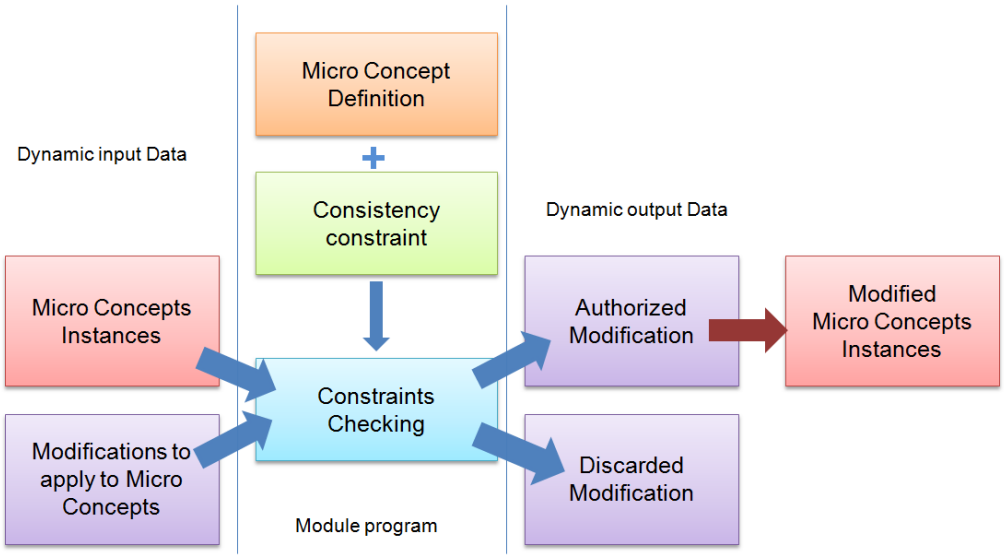

FIGURE A.2 – Processus de contrôle de la cohérence du modèle sémantique.

b) Module d'inférence

Il est en charge de l'exécution des règles, et de la synchronisation de la base de connaissances par rapport au modèle sémantique. Basé sur l'algorithme *Rete*, le module d'inférence est géré de manière synchrone avec le modèle sémantique. Afin de garantir la cohérence et l'intégrité du modèle sémantique, ce dernier ne peut pas être modifié par d'autres modules pendant le processus de raisonnement.

L'ontologie *AmiOnt* associée au langage de règles *SmartRules* est transformée en une représentation orientée objet grâce à la fonction de génération dynamique des classes de l'outil *ASM* [2]. Tous les concepts, propriétés, instances et actions sont instanciés sous forme de composants *JavaBeans*, et par conséquent, tous les faits insérés dans la mémoire de travail du moteur d'inférence *Drools* correspondent à des instances de classes Java.

Le moteur d'inférence *Drools* a été choisi à partir des considérations suivantes :
- *Drools* représente le moteur orienté objet open source le plus utilisé à l'heure actuelle ;
- *Drools* offre différents opérateurs de vérification de contraintes (*not matches, not contains, in, not*) ;
- *Drools* est un système orienté "règles de production" à la différence, par exemple, du moteur d'inférence *Jena*2. Ce dernier est une extension du mo-

2. http ://asm.ow2.org/index.html. Il est un outil de manipulation de classes Java conçu pour la génération et la manipulation dynamiques de code.

teur d'inférence *Jena*2, qui a été développé pour supporter le chainage avant. Cependant, l'algorithme *Rete* n'est pas totalement intégré dans le nouvelle version *Jena*2 ;
– *Drools* peut être considéré comme un bon candidat pour combiner les principes OWA et CWA. En effet, *Bragaglia et al.* [173] se sont attaqués au défi qui consiste combiner le raisonnement en logique de description $\mathcal{ALC}$ avec le raisonnement à base de règles. Les auteurs ont proposé une approche basée sur le moteur de règles *Drools* pour coupler les deux hypothèses CWA et OWA.

c) Module de traduction de règles

Il permet de traduire les règles *SmartRules* vers le format du moteur de règles *Drools*. La grammaire utilisée par ce module permet de construire automatiquement un analyseur qui détermine la structure syntaxique des règles. Cet analyseur permet de détecter des ambiguïtés syntaxiques à l'étape d'édition des règles.

d) Modules de communication et d'orchestration

Le noyau de raisonnement réactif comprend deux modules permettant d'une part, la communication entre les modules internes du noyau et d'autre part, l'orchestration de l'exécution de tous les modules. Le module de communication permet l'insertion dans le modèle sémantique des messages provenant des objets du monde réel, l'exécution des règles et la gestion de la persistance de l'état du système.

A.2 Le noyau de raisonnement narratif NKRL

Le rôle de ce noyau est d'inférer la causalité entre événements et d'établir la chronologie d'un contexte à partir d'événements passés et courants. Bien que prévu dans les spécifications, nous n'avons pas intégré les fonctions permettant de relier le module de communication et d'orchestration du noyau réactif avec celui du noyau du raisonnement narratif.

L'architecture du noyau $NKRL$, mise en œuvre initialement dans le cadre des projets Européens *Virthualise* et *Parmanides*, a été simplifiée dans le cadre de cette thèse. Elle est, dans sa version actuelle, composée principalement du module FUM, figure A.3. Ce module a pour rôle d'apparier (unifier) les occurrences prédicatives avec le motif recherché.

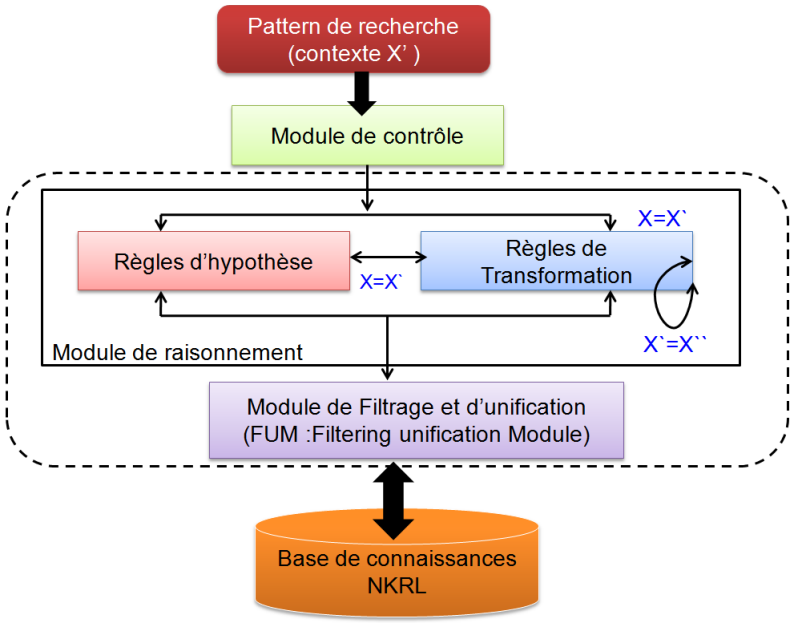

FIGURE A.3 – Architecture du mécanisme de raisonnement NKRL

Annexe B

Description de l'implémentation des scénarii

B.1 Implémentation des scénarii pour la validation du langage μConcept

La mise en œuvre des règles *SmartRules* pour la reconnaissance du contexte et l'adaptation du Système Intelligent Ambiant à ce dernier nécessite l'utilisation de la routine *instanceInternalUID* qui permet de générer un identifiant unique en l'associant à chaque instance ou action. Par exemple, l'instance du concept *FALL_SENSOR* représentée par l'instance *"http ://www.sembysem.org/AmiOnt#FALL_SENSOR"* est associée à une personne représentée par le symbole JOHN à l'aide de la propriété *wearedBy*. La propriété *isReferenceOf* permet, quant à elle, d'associer le détecteur d'ouverture/fermeture de porte à une porte donnant accès à la salle de séjour, représentée par l'instance *"http ://www.sembysem.org/AmiOnt#LIVING_ROOM_1"*. Par ailleurs, les routines *insert* et *update* permettent de mettre à jour les propriétés des instances ou des actions selon une approche de raisonnement non-monotone.

Dans ce qui suit, nous présentons l'implémentation des règles génériques pour la reconnaissance du contexte et l'adaptation du Système Intelligent Ambiant à ce dernier, dans le cadre des scénarii décrits dans le paragraphe 6.4.

B.1.1 Règles d'initialisation des scénarii

Les instances décrivant l'environnement d'expérimentation, les objets et les personnes impliquées sont créés en utilisant les règles ci-dessous.

Initialisation de l'environnement :

```
rule "init"
conditions
    not exists(House()) ;
actions
    House ?k1 := createInstance(House) ;
    ?k1 -> instanceInternalUID := "AmiOnt#KITCHEN_1" ;
    insert( ?k1) ;
    update( ?k1) ;
    House ?l1 := createInstance(House) ;
```

```
    ?l1 -> instanceInternalUID := "AmiOnt#LIVING_ROOM_1";
    insert(?l1);
    update(?l1);
    Pc ?pc1 := createInstance(Pc);
    ?pc1 -> instanceInternalUID := "AmiOnt#PC_1";
    insert(?pc1);
    ?pc1 -> hasLocation := ?l1;
    update(?pc1);
    Smart_Phone ?phone := createInstance(Smart_Phone);
    ?phone -> instanceInternalUID := "AmiOnt#SMART_PHONE_1";
    insert(?phone);
    update(?phone);
    Robot ?kompai := createInstance(Robot);
    ?kompai -> instanceInternalUID := "AmiOnt#ROBOT_KOMPAI";
    insert(?kompai);
    update(?kompai);
    Active_RFIDReader_Sensor ?reader := createInstance(Active_RFIDReader_Sensor);
    ?reader -> instanceInternalUID := "AmiOnt#LH1255_";
    insert(?reader);
    ?reader -> installedIn := ?pc1;
    update(?reader);
    Person ?john :=createInstance(Person);
    ?john -> instanceInternalUID := "AmiOnt#JOHN_";
    insert(?john);
    update(?john);
    Tag ?t1 :=createInstance(Tag);
    ?t1 -> instanceInternalUID := "AmiOnt#TAG_1";
    insert(?t1);
    ?t1 -> hasDataValue :="1022.0";
    ?t1 -> isReferenceOf := ?john;
    update(?t1);
    Localisation_Sensor ?cricket := createInstance(Localisation_Sensor);
    ?cricket -> instanceInternalUID := "AmiOnt#CRICKET_1";
    insert(?cricket);
    ?cricket -> wearedBy := ?john;
    update(?cricket);
    Fall_Sensor ?zcare := createInstance(Fall_Sensor);
    ?zcare -> instanceInternalUID := "AmiOnt#ZCARE_1";
    insert(?zcare);
    ?zcare -> wearedBy := ?john;
    update(?zcare);
    DoorStatus_Sensor ?zdoor := createInstance(DoorStatus_Sensor);
    ?zdoor -> instanceInternalUID := "AmiOnt#ZDOOR_1";
    insert(?zdoor);
    ?zdoor -> isReferenceOf := ?l1;
    update(?zdoor);
end
```

B.1.2 Règles de reconnaissance du contexte

La règle *ProximityObservation* est utilisée pour déduire une relation de proximité, exprimée par la propriété *nearTo* du concept *Person*.

```
rule "ProximityObservation"
conditions
    ?ac := Active_RFIDReader_Output( ?tagDetected :=one (hasValue), ?productor :=is-
ProducedBy,treated=="false") ;
    ?tagDetected( ?qualityValue := hasQualityValue) ;
    ?productor ( ?installed :=installedIn) ;
    ?tag :=Tag ( ?qualityValue == hasDataValue , ?person :=isReferenceOf) ;
actions
    ?person -> nearTo := ?installed ;
    update ( ?person) ;
    Proximity_Observation ?obs := createInstance(Proximity_Observation) ;
    ?obs->instanceInternalUID := "AmiOnt#PROXIMITY_OBSERVATION" ;
    insert( ?obs) ;
    ?obs->observedBy := ?productor ;
    ?obs->observationResult := ?ac ;
    update( ?obs) ;
    ?ac->treated :="true" ;
    update( ?ac) ;
end
```

La règle *PresenceObservation* permet d'inférer la présence d'une personne dans un espace donné, et par conséquent, que cet espace est occupé.

```
rule "PresenceObservation"
conditions
    ?presence := Location_Output( ?spacename := one (hasValue), ?productor :=isProduced-
By, treated=="false") ;
    ?spacename (isInstanceOf (Space_Name_Value), ?name :=hasQualityValue) ;
    ?space :=House( ?name == instanceInternalUID) ;
    ?productor ( ?person := wearedBy) ;
actions
    echo("******************** presence Detection **********************") ;
    ?person -> locatedAt := ?space ;
    update( ?person) ;
    Presence_Observation ?obspresence := createInstance(Presence_Observation) ;
    ?obspresence->instanceInternalUID := "AmiOnt#PRESENCE_OBSERVATION" ;
    insert( ?obspresence) ;
    ?obspresence->observedBy := ?productor ;
    ?obspresence->observationResult := ?presence ;
    update( ?obspresence) ;
    ?presence->treated :="true" ;
    update( ?presence) ;
end
```

La règle *LocationStatus* permet de déduire qu'un espace est accessible ou non.

```
rule "LocationStatus"
conditions
    ?doorstatus := DoorStatus_Output( ?spacename := one (hasValue), ?productor :=is-
ProducedBy, treated=="false") ;
    ?spacename (isInstanceOf (Space_Name_Value), ?name :=hasQualityValue) ;
    ?space :=House( ?name == instanceInternalUID, ?qualityValue == hasDataValue
, ?location :=isReferenceOf) ;
actions
    echo("******************** Accessible location **********************") ;
    ?person -> locatedAt := ?space ;
    update( ?person) ;
    Access_Observation ?obsaccess := createInstance(Access_Observation) ;
    ?obspresence->instanceInternalUID := "AmiOnt#ACCESS_OBSERVATION" ;
    insert( ?obsaccess) ;
    ?obsaccess ->observedBy := ?productor ;
    ?obsaccess ->observationResult := ? doorstatus ;
    update( ?obsaccess) ;
    ?doorstatus ->treated :="true" ;
    update( ?doorstatus) ;
end
```

B.1.3 Règles d'adaptation au contexte

L'adaptation au contexte est décrite par la règle *HealthAdvice*. Cette dernière consiste à émettre des conseils de santé (prise de médicament, conseils diététiques) en utilisant le moyen de communication le plus proche de la personne. La règle *ProximityObservation* permet d'inférer que la personne est proche de son ordinateur. L'ordinateur de la personne est ainsi utilisé ici comme moyen de communication pour émettre le conseil médical.

```
rule "HealthAdvice"
conditions
    ?health :=Health_Advice_Output() ;
    Proximity_Observation( ?observedby :=observedBy) ;
    ?observedby( ?location :=installedIn) ;
    Person(nearTo== ?location) ;
actions
    echo("———- Remember to take medicine using PC————————") ;
    _VisualMessage ?message := createAction( ?location, _VisualMessage) ;
    ?message->content :="You must take your drug" ;
    execute( ?message) ;
    removeInstance( ?health) ;
end
```

Cette règle est utilisée pour émettre un conseil médical en utilisant le robot.

La règle ci-dessous permet de déplacer le robot vers l'endroit où se trouve la personne.

```
rule "HealthAdviceRobot" conditions
    ?health :=Health_Advice_Output() ;
    Presence_Observation( ?observedby :=observedBy, ?observation :=observationResult) ;
    ?observation( ?xValue := one (hasValue), ?yValue :=one (hasValue) ) ;
    ?xValue (isInstanceOf (XValue), ?x :=hasQuantityValue) ;
    ?yValue (isInstanceOf (YValue), ?y :=hasQuantityValue) ;
    ?rob :=Robot() ;
actions
    echo("———- Remember to take medicine using Robot————————") ;
    _MoveRobot ?moverobot := createAction( ?rob, _MoveRobot) ;
    ?moverobot-> content := "John ! You must take your drug" ;
    ?moverobot-> x := ?x ;
    ?moverobot-> y := ?y ;
    execute( ?moverobot) ;
    removeInstance( ?health) ;
end
```

La gestion d'une situation d'urgence due à une chute consiste ici à procéder à une levée de doute après le déclenchement de l'alarme correspondante. Pour ce faire, nous utilisons les règles suivantes : *CheckPersonStatus*, *EmergencyButtonPushing*, *EmergencyAlarm*. Une fois le robot arrivé à l'endroit où la personne est localisée, il vérifie si cette dernière est consciente. Il s'agit ici d'un contexte non-observable déduit à partir d'un dialogue établi entre le robot et la personne. Si cette dernière n'interagit pas avec le robot, elle est considérée comme inconsciente et par conséquent, le contexte courant correspond à une situation d'urgence. L'adaptation à ce contexte, réalisée à l'aide de la règle *EmergencyButtonPushing*, consiste alors à déclencher une alarme en envoyant un message d'alerte sur un Smartphone.

```
rule "CheckPersonStatus"
conditions
    ?fall :=Fall_Output(treated=="false") ;
    Presence_Observation( ?observedby :=observedBy, ?observation :=observationResult) ;
    ?observation( ?xValue := one (hasValue), ?yValue :=one (hasValue) ) ;
    ?xValue (isInstanceOf (XValue), ?x :=hasQuantityValue) ;
    ?yValue (isInstanceOf (YValue), ?y :=hasQuantityValue) ;
    ?rob :=Robot() ;
actions
    echo("———- Check Person Status————————") ;
    _MoveRobot ?moverobot := createAction( ?rob, _MoveRobot) ;
    ?moverobot-> content := "John ! If you need assistance, push the SOS button" ;
    ?moverobot-> x := ?x ;
    ?moverobot-> y := ?y ;
    execute( ?moverobot) ;
    ?fall-> treated :="true" ;
    update( ?fall) ;
end
```

```
rule "EmergencyButtonPushing"
conditions
    ?ihr := IHR_Output( ?buttonname := one (hasValue), treated=="false") ;
    ?buttonname ( ?name :=hasQualityValue) ;
actions
    echo("------- Button clicked ----------") ;
    Button_Pushing ?button := createInstance(Button_Pushing) ;
    ?button->instanceInternalUID := "AmiOntO#EMERGENCY_BUTTON" ;
    insert( ?button) ;
    update( ?button) ;
    ?ihr-> treated :="true" ;
    update( ?ihr) ;
end

rule "EmergencyAlarm"
conditions
    ?fall :=Fall_Output() ;
    ?ihr :=IHR_Output() ;
    not exists ( Button_Pushing()) ;
    ?smart :=Smart_Phone() ;
    ?person :=Person( ?location :=locatedAt) ;
actions
    echo("------- Emrgency Alarm ----------") ;
    _EmergencyAlarm ?alarm := createAction( ?smart, _EmergencyAlarm) ;
    ?alarm->content :="Emergency" ;
    ?alarm->humanLocalization := ?location->instanceInternalUID ;
    execute( ?alarm) ;
    removeInstance( ?fall) ;
    removeInstance( ?ihr) ;
end
```

B.2 Implémentation des scénarii du modèle NKRL

Dans ce paragraphe, nous présentons en détail les gabarits des règles d'hypothèses et de transformations NKRL utilisées pour la reconnaissance du contexte non-observable "Situation urgente", que nous avions mise en œuvre. Pour ce faire, nous rappelons ici que le symbole *NLDescription* donne une description en langage naturel d'un template (gabarit). Par ailleurs, un rôle ou une variable définie entre crochets ([]) est considéré(e) comme optionnel(le), tandis que les autres (variables ou rôles) sont obligatoires. Les variables *var1, ... ,var10* représentent des contraintes permettant de vérifier que les valeurs assignées à chaque variable lors de la création d'une occurrence de prédicat sont spécifiques aux termes (concept, instances) utilisés dans le composant définitionnel. Ainsi, les contraintes définies dans chaque template de l'ontologie *HTemp* sont associées aux concepts définis dans l'ontologie *HClass*.

NLDescription : 'Creation of Generic Entities, Elements, Ornaments etc."
PREDICATE : PRODUCE
SUBJ var1 : [(var2)]
OBJ var3
[SOURCE var4 : [(var5)]]
[BENF var6 : [(var7)]]
[MODAL var8]
[TOPIC var9]
[CONTEXT var10]

{[modulators] !=abs}
{[temporal attributes]}
var1 = <human_being_or_social_body>
var2 = <location_> | <pseudo_sortal_geographical>
var3 = <artefact_> | <information_content> | <internet_location>
var4 = <human_being_or_social_body>
var5 = <location_> | <pseudo_sortal_geographical>
var6 = <human_being_or_social_body>
var7 = <location_> | <pseudo_sortal_geographical>
var8 = <artefact_> | <activity_> | <process_> | <temporal_development>
var9 = <physical_appearance> | <situation_>
var10 = <situation_> | <symbolic_label>

TABLE B.1 – Structure du template PRODUCE

B.2.1 Gabarit PRODUCE

Ce gabarit est utilisé dans la requête pour déterminer l'initiateur de l'événement correspondant au déclenchement de l'alarme, tableau B.1. Il est également utilisé dans la condition 3 de la règle d'hypothèse visant à apporter des précisions sur le contenu du message émis par le robot à l'attention de JOHN_.

B.2.2 Gabarit OWN : CONTROL

Ce gabarit est utilisé dans la condition 1 de la règle d'hypothèse et dans l'antécédent de la règle de transformation 1, tableau B.2.

B.2.3 Gabarit OWN : CompoundProperty

Ce gabarit est utilisé dans la condition 4 de la règle d'hypothèse et dans le conséquent 2 de la règle de transformation 1, tableau B.3.

B.2.4 Gabarit Behave

Ce gabarit est utilisé dans l'antécédent de la règle de transformation 2, tableau B.4.

NLDescription : 'Be in Control of Someone/Something'."
PREDICATE : OWN
SUBJ var1 : [(var2)]
OBJ var3
[SOURCE var4 : [(var5)]]
[BENF var6 : [(var7)]]
[MODAL var8]
TOPIC var9
[CONTEXT var10]

{[modulators]!=abs}
{[temporal attributes]}
var1 = <country_> | <human_being_or_social_body>
var2 = <location_> | <pseudo_sortal_geographical>
var3 = <control_>
var4 = <country_> | <human_being_or_social_body>
var5 = <location_> | <pseudo_sortal_geographical>
var6 = <country_> | <human_being_or_social_body>
var7 = <location_> | <pseudo_sortal_geographical>
var8 = <activity_> | <process_> | <war_>
var9 = <country_> | <human_being_or_social_body> | <market_sector>
var10 = <situation_> | <symbolic_label>

Table B.2 – Structure du template OWN : CONTROL

NLDescription : ' Behave Templates'
PREDICATE : BEHAVE
SUBJ var1 : [(var2)]
OBJ var3 : [(var4)]
[SOURCE var5 : [(var6)]]
[BENF var7 : [(var8)]]
[MODAL var9]
[TOPIC var10]
[CONTEXT var11]

{[modulators]!=abs}
{[temporal attributes]}
var1 = <human_being_or_social_body>
var2 = <location_> | <pseudo_sortal_geographical>
var3 = <sortal_concept>
var4 = <location_> | <pseudo_sortal_geographical>
var5 = <human_being_or_social_body>
var6 = <location_> | <pseudo_sortal_geographical>
var7 = <human_being_or_social_body>
var8 = <location_> | <pseudo_sortal_geographical>
var9 = <h_class>
var10 = <sortal_concept>
var11 = <situation_> | <symbolic_label>

Table B.4 – Structure du template BEHAVE

NLDescription : 'The Property is Necessarily Represented by a SPECIF(ication) List Filling the TOPIC Slot'
PREDICATE : OWN
SUBJ var1 : [(var2)]
OBJ var3
[SOURCE var4 : [(var5)]]
!($BENF$)
[MODAL var6]
$TOPIC$ ($SPECIF var7$ $var8$)
[CONTEXT var9]

{[modulators] !=abs}
{[temporal attributes]}
var1 != <human_being_or_social_body> | <property_>
var2 = <location_> | <pseudo_sortal_geographical>
var3 = <property_>
var4 = <human_being_or_social_body>
var5 = <location_> | <pseudo_sortal_geographical>
var6 = <activity_> | <artefact_> | <process_> | <pseudo_sortal_concept> | <reified_event> | <symbolic_label> | <temporal_sequence>
var7 = <part/whole_relationship> | <relational_property> | <spatio/temporal_relationship>
var8 = <sortal_concept>
var9 = <situation_> | <symbolic_label>

TABLE B.3 – Structure du template OWN : CompoundProperty

Bibliographie

[1] Georgia Institute of Technology, "from research to Market : Smart Shirt Moves", *Georgia Institute of Technology, GA, USA*, 2004. (Cité en page 9.)

[2] J.F. Allen. "An Interval-Based Representation of Temporal Knowledge", In *Proceedings of the 7th international joint conference on Artificial intelligence*, volume 1, IJCAI'81, pages 221–226, Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 1981. (Cité en page 58.)

[3] J.F. Allen. "Maintaining knowledge about temporal intervals", In *ACM*, volume 26, pages 832–843, New York, NY, USA, 1983. (Cité en page 58.)

[4] J.F. Allen. "Towards a General Theory of Action and Time", In *Artif. Intell. Elsevier Science Publishers Ltd*, volume 23, pages 123–154, Essex, UK, 1984. (Cité en page 58.)

[5] L. Aogán, D. Dermot, and L. Matt. "Point-of-need diagnosis of cystic fibrosis using a potentiometric ion-selective electrode array", In *Analyst*, volume 125, pages 2264–2267, 2000. (Cité en page 9.)

[6] A. Artale and E. Franconi. "A Survey of Temporal Extensions of Description Logics", In *Annals of Mathematics and Artificial Intelligence*, volulme 30, pages 171–210, 2000. (Cité en page 62.)

[7] F. Baader and B. Hollunder. "KRIS : Knowledge Representation and Inference System", In *SIGART Bull.*, volume 2, pages 8–14, 1991. (Cité en page 34.)

[8] J.F. Baget and M.L. Mugnier. "The SG family : extensions of simple conceptual graphs", In *Proceedings of the 17th international joint conference on Artificial intelligence*, Volume 1, IJCAI, pages 205–210, San Francisco, CA, USA, 2001. (Cité en page 30.)

[9] J.F. Baget. "Extending the CG Model by Simulations", In *Conceptual Structures : Logical, Linguistic, and Computational Issues Lecture Notes in Computer Science,* volulme 1867, pages 277–291, 2000. (Cité en pages xiii et 33.)

[10] J.F. Baget and M.L. Mugnier. "Extensions of Simple Conceptual Graphs : The Complexity of Rules and Constraints" In *Journal Of Artificial Intelligence Research*, volume 16, pages 425-465, 2002. (Cité en page 33.)

[11] M. Bal. *"Narratology : Introduction to the Theory of Narrative,"*. 2 nd edn. Toronto : University of Toronto Press., 1997. (Cité en page 92.)

[12] M. Baldauf, S. Dustdar, and F. Rosenberg. "A survey on context :aware systems", In *Journal International Journal of Ad Hoc and Ubiquitous Computing*,volume 2, pages 263–277, Geneva, SWITZERLAND 2007. (Cité en page 41.)

[13] S. Barry and C.A. Welty. "FOIS introduction : Ontology- - -towards a new synthesis", In *Proceedings of the international conference on Formal Ontology*

in Information Systems, volume 2001, pages 3–9, New York, USA, 2001. (Cité en page 27.)

[14] S. Batsakis and E.G. M. Petrakis. "SOWL : A Framework for Handling Spatio-Temporal Information in OWL 2.0", *Proceedings of the 5th international conference on Rule-based reasoning, programming, and applications, RuleML'2011*, pages 242–249, Berlin, Heidelberg, 2011. (Cité en page 63.)

[15] M. Beetz, L. Mösenlechner, and M. Tenorth. "CRAM - A Cognitive Robot Abstract Machine for Everyday Manipulation in Human Environments", In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference*, pages 1012–1017, 2010. (Cité en pages xiii et 53.)

[16] T. Berners-Lee, R. Fielding, and L. Masinter. "Uniform Resource Identifiers (URI) : Generic Syntax", Technical report, 1998. (Cité en page 37.)

[17] T.R. Gruber, "Ontolingua : A Mechanism to Support Portable Ontologies", Technical report, 1992. (Cité en page 28.)

[18] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, and D. Riboni. "A survey of context modelling and reasoning techniques", In *Pervasive Mob. Comput. Elsevier Science Publishers B. V.*, volume 6, pages 161–180, 2010. (Cité en page 41.)

[19] R.J. Brachman, J.G. Schmolze. "An overview of the KL-ONE Knowledge Representation System", In *Journal of Cognitive Science*, volume 9, pages 171–216, 1985. (Cité en page 28.)

[20] J. Bohn, V. Coroama, M. Langheinrich, F. Mattern, and M. Rohs. "Living in a World of Smart Everyday Objects-Social, Economic, and Ethical Implications", In *Journal of Human and Ecological Risk Assessment*, volume 5, pages 763–786, 2004. (Cité en page 7.)

[21] R.J. Brachman. "What's in a concept : Structural foundations for semantic networks", In *Int. Journal of Man-Machine Studies*, volume 9, pages 127–152, 1977. (Cité en page 33.)

[22] R.J. Brachman. "Structured Inheritance Networks", In *Int. Journal of Man-Machine Studies*, volume 9, pages 127–152, 1978. (Cité en page 33.)

[23] R.J. Brachman and J.G. Schmolze. "An Overview of the KL-ONE Knowledge Representation System", In *Cognitive Science*, volume 9, pages 171–216, 1985. (Cité en pages 34 et 48.)

[24] D. Brickley and R.V. Guha. "W3C Recommendation. World Wide Web Consortium", 2004. Available at `http://www.w3.org/TR/2000/CR-rdf-schema-20000327/`. (Cité en page 38.)

[25] Available at `http:/www.hitech-projects.com/euprojects/amigo`. (Cité en page 47.)

[26] M. Chein, M.L. Mugnier, "Conceptual Graphs : fundamental notions". *Revue d'Intelligence Artificielle*, volume 6, pages 365–406, 1992. (Cité en pages 30 et 33.)

[27] M. Chein and M.L. Mugnier. “Conceptual Graph are also Graph”, Technical report, 1995. (Cité en page 33.)

[28] M. Chein and M.L. Mugnier. “Positive Nested Conceptual Graphs”, In *Proc. 5th Int Conf. on Conceptual Structures (ICCS 97), Springer Verlag, LNAI 1257*, pages 95–109, 1997. (Cité en page 30.)

[29] H. Chen, T. Finin and A. Joshi. “A Context Broker for Building Smart Meeting Rooms”, In *AAAI 2004 Spring Symposium on Knowledge Representation and Ontology for Autonomous Systems*, Stanford, 2004. (Cité en page 47.)

[30] H. Chen, T. Finin and A. Joshi. “The SOUPA Ontology for Pervasive Computing”, In *Ontologies for Agents : Theory and Experiences*, pages 233–258. BirkHauser, 2005. (Cité en page 44.)

[31] D.J. Cook and S.K. Das. “How Smart are our Environments ? An Updated Look at the State of the art”, In *Journal of Pervasive and Mobile Computing*, volume 3, pages 53–73. In : , 2007. (Cité en page 8.)

[32] D. Davidson. “Causal Relations”, In *The Journal of Philosophy*,volume 64, pages 691–703, 1967. (Cité en page 108.)

[33] G. Demiris and J. Tan. “rejuvenating Home Helath Care and Tele-Homecare”, *In : J. Tan (Ed.) : E-Health Care Information Systems : An Introduction for Students and Professionals. Jossey-Bass, San Francisco, CA, USA.*, pages 267–290, 2005. (Cité en page 9.)

[34] E. Demmester, A. Huntemann, D. Vanhooydonck, G. Vanacker, A. Deggest, H. Van Brussel, M. Nuttin. “Bayesian estimation of wheelchair driver intents : Modeling intents as geometric paths tracked by the driver”, *IEEE/RSJ Int Conf on Intelligent Robots and Systems (IROS)*, pages 5775–5780, 2006. (Cité en page 14.)

[35] G.D. Abowd, A.K. Dey, P.J. Brown, N. Davies,M. Smith, P. Steggles. “Towards a Better Understanding of Context and Context-Awareness”, In *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pages 304–307, Springer-Verlag, 1999. (Cité en page 19.)

[36] T. Deyle, H. Nguyen, M.S. Reynolds, and C.C. Kemp. “RFID-Guided Robots for Pervasive Automation”, *Pervasive Computing, IEEE*, volume 9, pages 37–45, 2010. (Cité en pages xiii, 10 et 11.)

[37] P. Di, J. Huang, K. Sekiyama, and T. Fukuda. “Motion control of intelligent cane robot under normal and abnormal walking condition”, In *RO-MAN, IEEE*, pages 497–502, 2011. (Cité en page 14.)

[38] K. Ducatel, M. Bogdanowicz, F. Scapolo, J. Leijten, and J.C Burgelman. “Scenarios of Ambient Intelligence”, *Final Report,* 2001. (Cité en pages 6 et 7.)

[39] R. Fikes, P. Hayes, and I. Horrocks. “OWL-QL- A Language for Deductive Query Answering on the Semantic Web”, *Web Semant*, volume 2, pages 19–29, Elsevier Science Publishers B. V, Amsterdam, 2004. (Cité en page 40.)

[40] Charles L. Forgy. "Rete : A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem", In *Artificial Intelligence*, volume 19, pages 17–37, 1982. (Cité en page 76.)

[41] C. Galindo, J.A. Fernandez-Madrigal, J. Gonzalez, and A. Gonzalez. "Using Semantic Information for Improving Efficiency of Robot Task Planning", Rome, Italy, 2007. (Cité en pages xiii et 43.)

[42] V. Haarslev, R. Möller, M. Wessel. "Querying the Semantic Web with Racer + nRQL", In *Proceedings of the KI-2004 International Workshop on ADL'04,*, 2004. (Cité en page 40.)

[43] M. Gandy, T. Starner, J. Auxier, and D. Ashbrook. "The Gesture Pendant : A Self-Illuminating, Wearable, Infrared Computer Vision System for Home Automation Control and Medical Monitoring", In *Proceedings of the 4th IEEE International Symposium on Wearable Computers*, pages 87–94, Washington, DC, USA, 2000. IEEE Computer Society. (Cité en page 8.)

[44] J.J Gibson. "The Ecological Approach to Visual Perception", *Lawrence Erlbaum Associates*, 1979. (Cité en page 43.)

[45] P.A. Gómez, F-L. Mariano, and O. Corcho. "Ontological Engineering, with examples from the areas of knowledge", pages 3–15. Management, e-Commerce and the Semantic Web, Springer, 2004. (Cité en page 28.)

[46] C. Gopalsamy, S. Park, R. Rajamanickam, and S. Jayaraman. "The Wearable Motherboard : The first Generation of adaptive and responsive textile structures (arts) for medical applications", *Virtual Reality*, volume 4, pages 152–168, 1999. (Cité en page 9.)

[47] T.R. Gruber. "Toward Principles for the Design of Ontologies Used for Knowledge Sharing", In *International Journal of Human-Computer Studies*, pages 907–928. Kluwer Academic Publishers, 1993. (Cité en page 27.)

[48] H. Guoqiang, P.T. Wee, and W. Yonggang. "Cloud robotics : architecture, challenges and applications", *Network, IEEE*, volume 26, pages 21–28, 2012. (Cité en page 17.)

[49] C. Gutierrez, C.A. Hurtado, and A. Vaisman. "Introducing Time into RDF", *Knowledge and Data Engineering, IEEE Transactions*, volume 19, pages 207 – 218, 2007. (Cité en page 61.)

[50] N. Halin, M. Junnila, P. Loula, and P. Aarnio. "The LifeShirt system for wireless patient monitoring in the operating room", *Journal of Telemedicine and Telecare.*, volume 11, pages 41–43, 2005. (Cité en page 9.)

[51] P. Hallot. "Relations Spatio-Temporelles dans un espace-temps primitif. un essai de simplification de l'analyse Spatio-Temporelle", Master's thesis, Université de Liège. Faculté des Sciences géographiques, 2006. (Cité en pages xiii et 58.)

[52] F.V. Harmelen, V. Lifschitz, and B. Porter. "Handbook of Knowledge Representation, 1st Edition", Elsevier Science, 2008. (Cité en page 38.)

[53] S.S. Hidayat, B.K. Kim, and K. Ohba. "Learning Affordance for Semantic Robots Using Ontology Approach", In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference*, pages 2630 –2636, sept. 2008. (Cité en page 43.)

[54] J. Higginbotham. "The Logic of Perceptual Reports : An Extensional Alternative to Situation Semantics", *Journal of Philosophy*, volume 80, pages 100–127, 1983. (Cité en page 108.)

[55] I. Horrocks. "OWL Rules, OK ?", In *Rule Languages for Interoperability*, 2005. (Cité en page 41.)

[56] I. Horrocks and P. F. Patel-Schneider. "KR and reasoning on the semantic web : OWL", In J. Domingue, D. Fensel, and J.A. Hendler, editors, *Handbook of Semantic Web Technologies*, chapter 9, pages 365–398. Springer, 2011. (Cité en page 40.)

[57] I. Horrocks, P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. "SWRL : A Semantic Web Rule Language combining OWL and RuleML", 2004. Available at `http://www.w3.org/Submission/SWRL/`. (Cité en page 41.)

[58] H. Jagos, J. Oberzaucher, and W. L. Zagler. "Erste Schritte bei der Entwicklung Instrumentierter Schuhe zur sturzvorbeugung alter menschen", *IKTForum*, 2007. (Cité en page 9.)

[59] K. Jong-Hwan, J. In-Bae, P. In-Won, and L. Kang-Hee. "Multi-Layer Architecture of Ubiquitous Robot System for Integrated Services", *I. J. Social Robotics*, volume 1, pages 19–28, 2009. (Cité en page 48.)

[60] K. Kamei, S. Nishio, N. Hagita, and M. Sato. "Cloud networked robotics", *Network, IEEE*, volume 26, pages 28 –34, 2012. (Cité en page 17.)

[61] R. Kikin-Gil. "BuddyBeads : techno-jewelry for non-verbal communication within teenager girls groups," *Personal Ubiquitous Comput.*, volume 2, pages 106–109, 2006. (Cité en page 8.)

[62] J. H. Kim, Y.D. Kim, and K.H. Lee. "The Third Generation of Robotics : Ubiquitous Robot", *In Proc. of the 2nd Int. Conf. on Autonomous Robots and Agents, Palmerston North, New Zealand*, 2004. (Cité en page 16.)

[63] R. Kowalski and M. Sergot. "A Logic-Based Calculus of Events", *New Gen. Comput. Ohmsha*, volume 4, pages 67–95, January 1986. (Cité en page 60.)

[64] M. Krötzsch, F. Maier, A. Maier, and P. Hitzler. "A Better Uncle for OWL : Nominal schemas for integrating rules and ontologies", In *Proceedings of the 20th International Conference on World Wide Web*, pages 645–654, ACM, 2011. (Cité en page 41.)

[65] J. A. Kuffner. "A Crowd of Quantum", *Ieee Spectrum 48*, pages 16–18, 2011. (Cité en page 17.)

[66] F. Landman. *"Events and Plurality :The Jerusalem Lectures (Studies in Linguistics and Philosophy"*, Kluwer Academic Publisher, 2000. (Cité en page 108.)

[67] O. Lassila and D. Khushraj. "Contextualizing applications via semantic middleware", MOBIQUITOUS, pages 183–191, Washington, DC, USA, 2005. (Cité en page 41.)

[68] J. Legon. "'Smart sofa' aimed at couch potatoes", pages 217–236. In : CNN, 2003. (Cité en page 8.)

[69] F.W. Lehmann and E.Y. Rodin. *"Semantic networks in artificial intelligence"*, volume 12 of *International series in modern applied mathematics and computer science*. Pergamon Press, 1992. (Cité en page 29.)

[70] S. Lemaignan, R. Ros, L. Mosenlechner, R. Alami, and M. Beetz. "ORO, a knowledge management platform for cognitive architectures in robotics", In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference*, pages 3548–3553, 2010. (Cité en pages xiii, 49, 50 et 51.)

[71] S. Leonhardt. "Personal Healthcare Devices," In S. Mukherjee, R. Aarts, R. Roovers, F. Widdershoven, and M. Ouwerkerk, editors, *"AmIware Hardware Technology Drivers of Ambient Intelligence,"*, volume 5 of *Philips Research*, pages 349–370. Springer Netherlands, 2006. (Cité en page 9.)

[72] H.J. Levesque, F. Pirri, and R. Reiter. "Foundations for the Situation Calculus", *Electron. Trans. Artif. Intell*, volume 2, pages 159–178, 1998. (Cité en page 59.)

[73] L. Sabri, A. Chibani, Y. Amirat, and G.P Zarri. "Semantic Reasoning Framework to Supervise and Manage Contexts and Objects in Pervasive Computing Environments", In *Proceedings of the 2011 IEEE Workshops of International Conference on Advanced Information Networking and Applications, IEEE Computer Society*, WAINA '11, Washington, DC, USA, pages 47–52, Washington, DC, USA, 2011. IEEE Computer Society. (Cité en pages 48 et 119.)

[74] J.S. Brunner, J.F. Goudou, P. Gatellier, J. Beck, and C.E. Laporte. "SEMbySEM : a framework for sensor management", In *Proc. of the 1st Int. Workshop on the Semantic Sensor Web (SemSensWeb)*, 2009. (Cité en page 119.)

[75] D. Fensel, I. Horrocks, F. Harmelen, S. Decker, M. Erdmann, M. Klein, "OIL in a Nutshell", In R. Dieng, O. Corby, editors, *Knowledge Engineering and Knowledge Management Methods, Models, and Tools*, volume 1937, pages 1–16. Spring Berlin Heidelberg, 2000. (Cité en page 39.)

[76] M.R. MacGregor, "Inside the LOOM Description Classifier", In *Journal of SIGART Bull*, volume 2, pages 88–92. ACM, 1991. (Cité en page 28.)

[77] J. Liberman. "Things That Think", pages 217–236. MIT press, Massachusetts Institute of Technology, Cambridge, MA, USA, 2006. (Cité en page 8.)

[78] I. Horrocks. "DAML+OIL : A Reason-able Web Ontology Language", In C. Jensen, S. Šaltenis, K. Jeffery, J.Pokorny, E. Bertino, K. Böhn, M. Jarke, editors, *Advances in Database Technology - EDBT*, volume 2287, pages 2–13. Spring Berlin Heidelberg, 2002. (Cité en page 28.)

[79] V. Lifschitz. "Circumscription", In D.M. Gabbay, C.J. Hogger, and J.A. Robinson, editors, *Handbook of logic in artificial intelligence and logic programming*, volume 3, pages 297–352. Oxford University Press, Inc., New York, NY, USA, 1994. (Cité en page 60.)

[80] C. Lutz, F. Wolter, and M. Zakharyashev. "Temporal Description Logics : A Survey", *Temporal Representation and Reasoning. TIME'08. 15th International Symposium*, pages 3–14, 2008. (Cité en page 62.)

[81] R.M. MacGregor. "Inside the LOOM description classifier", *SIGART Bull. ACM*, volume 2, pages 88–92, 1991. (Cité en page 34.)

[82] J. Manfred. "Narratology : A Guide to the Theory of Narrative", English Department, University of Cologne., 2005. Available at `http://www.uni-koeln.de/~ame02/pppn.htm`. (Cité en page 92.)

[83] J. McCarthy. "Circumscription- A Form of non-monotonic reasoning", In *Artificial Intelligence*, 1980. Conference Chair-Smith, Barry and Conference Chair-Welty, Christopher. (Cité en pages 27 et 59.)

[84] J. McCarthy, M.L. Minsky, N. Rochester, and C.E. Shannon. "A proposal for the dartmouth summer research project on artificial intelligence", 1955. (Cité en page 48.)

[85] J. McCarthy and P.J. Hayes. "Some philosophical problems from the standpoint of artificial intelligence", In in B. Meltzer and D. Michie, editors, *Machine Intelligence* , volume 4, pages 463–502. Edinburgh University Press, Edinburgh, 1969. (Cité en page 59.)

[86] D. V. McDermott. "A Temporal Logic for Reasoning About Processes and Plans", *Cognitive Science*, volume 6, pages 101–155, 1982. (Cité en page 58.)

[87] G. Meditskos and N. Bassiliades. "A Rule-Based Object-Oriented OWL Reasoner", *IEEE Trans. on Knowl. and Data Eng.*, volume 20, pages 397–410, 2008. (Cité en page 41.)

[88] G. Meditskos and N. Bassiliades. "CLIPS-OWL : A framework for providing object-oriented extensional ontology queries in a production rule engine", *Data Knowl. Eng.*, volume 70, pages 661–681, 2011. (Cité en page 41.)

[89] S. Mefoued, S. Mohammed, and Y. Amirat. "Knee joint movement assistance through robust control of an actuated orthosis", *IEEE International Conference on Intelligent and Robotic Systems (IROS), San Francisco USA*, pages 1749–1754, 2011. (Cité en page 14.)

[90] R. Miller and M. Shanahan. "Some Alternative Formulations of the Event Calculus", In *Computational Logic : Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part II*, pages 452–490, London, UK, UK, 2002. Springer-Verlag. (Cité en page 60.)

[91] M. Minsky. "A framework for representing knowledge", Technical report, 1974. (Cité en page 29.)

[92] R. Montague. "The proper treatment of quantification in ordinary english," In J. Hintikka, J. Moravcsik, and P. Suppes, editors, *Approaches to Natural Language : proceedings of the 1970 Stanford workshop on Grammar and Semantics*, pages 221–242. Reidel, Dordrecht, 1973. (Cité en page 108.)

[93] B. Mordechai. "Mathematical Logic for computer science", *Prentice-Hall, Englewood Cliffs*, pages 11–87, 1993. (Cité en page 26.)

[94] M.L. Mugnier. "Knowledge Representation and Reasonings Based on Graph Homomorphism", In *Proc. ICCS, volume 1867 of LNAI*, pages 172–192. Springer, 2000. (Cité en page 30.)

[95] B.J. Munro, J.R. Steele, T.E. Campbell, and G.G. Wallace. "Wearable eHealth Systems for Personalised Health Management : state of the art and future challenges", *Lymberis, Andreas, IOS Press, Amsterdam*, pages 271–277, 2004. (Cité en page 9.)

[96] C. Nieto-Granda, J.G. Rogers, A.J.B Trevor, and H.I Christensen. "Semantic map partitioning in indoor environments using regional analysis", In *International Conference on Intelligent RObots and Systems - IROS*, pages 1451–1456, 2010. (Cité en page 48.)

[97] A. Nüchter and J. Hertzberg. "Towards semantic maps for mobile robots", *Robot. Auton. Syst.*, 56(11) :915–926, 2008. (Cité en pages 19 et 48.)

[98] P. S. Pandian, K. P. Safeer, G. Pragati, D. T. Shakunthala, B. S. Sundersheshu, and V. C. Padaki. "Wireless sensor network for wearable physiological monitoring", volume 3, pages 21–29, 2008. (Cité en page 9.)

[99] P.S. Pandian, K. Mohanavelu, K.P. Safeer, T.M. Kotresh, D.T. Shakunthala, G. Parvati, and V.C. Padaki. "Smart Vest : Wearable multi-parameter remote physiological monitoring system", 30 :466–477, 2008. (Cité en page 9.)

[100] S.H. Park, S.H. Won, J.B. Lee, and S.W. Kim. "Smart home - digitally engineered domestic life", volume 7, pages 189–196. In : Personal Ubiquitous Computing, 2003. (Cité en page 8.)

[101] T. Parsons. "Events in the Semantics of English : A Study in Subatomic Semantics, Current Studies in Linguistics series", Technical report, MIT Press., 1990. (Cité en page 108.)

[102] C. Patel, J. Cimino, J. Dolby, A. Fokoue, A. Kalyanpur, A. Kershenbaum, L. Ma, E. Schonberg, and K. Srinivas. "Matching patient records to clinical trials using ontologies", In *Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference*, ISWC'07/ASWC'07, pages 816–829, Berlin, Heidelberg, 2007. Springer-Verlag. (Cité en page 42.)

[103] P. F. Patel-Schneider, L. A. Resnick, D. L. McGuinness, E. Weixelbaum, M. Abrahams, and A. Borgida. "Neoclassic user's guide : Version 1.0.", Artificial Intelligence Principles Research Department, AT&T Labs Research, 1996. (Cité en page 44.)

[104] Peter F. Patel-Schneider, Deborah L. McGuinness, Ronald J. Brachman, and Lori Alperin Resnick. "The classic Knowledge Representation System : Guiding Principles and Implementation Rationale", *SIGART Bull.*, 2(3) :108–113, 1991. (Cité en page 44.)

[105] M. Perry, P. Jain, and A.P. Sheth. "SPARQL-ST : Extending SPARQL to Support Spatiotemporal Queries", *Geospatial Semantics and the Semantic Web, edited by, Ashish, N. and Sheth, A.P.*, volume 12, pages 61–86, 2011. (Cité en page 62.)

[106] D. J. "Philips and " Das, S.K. "Vision of the Future", *Philips Corporate Design, Eindhoven, The Netherlands.*, 2006. (Cité en page 8.)

[107] L. Sabri, A. Chibani, T. Patkos, F. Sebbak, and Y. Amirat. "Advanced Context modeling and Reasoning for ambient intelligence environments and ubiquitous robotics", In *in : Journées Nationales de la Robotique Interactive du GDR Robotique*, JNRI 2011, Paris, France, 2011. (Cité en page 48.)

[108] H. Pigot, J. Bauchet, and S. Giroux. "Assistive Devices for People with Cognitive Impairments", In *The Engineering Handbook of Smart Technology for Aging, Disability, and Independence*, chapter 12, pages 217–236. John Wiley & Sons, Inc., 2008. (Cité en pages 8 et 19.)

[109] N.B. Priyantha. *"The Cricket Indoor Location System,"* PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 2005. (Cité en page 12.)

[110] J. Pustejovsky, J/M. Castaño, R. Ingria, R. Sauri, R.J. Gaizauskas, A. Setzer, G. Katz, and D.R. Radev. "TimeML : Robust Specification of Event and Temporal Expressions in text", In *New Directions in Question Answering*, pages 28–34, 2003. (Cité en page 61.)

[111] M.R. Quillian. "Word Concepts : A theory and simulation of some basic semantic capabilities", *Behavioral Science*, volume 12, pages 410–430, 1967. (Cité en page 29.)

[112] P. Reignier. *"Intelligence Ambiante Pro-Active de la Spécification à l'Implémentation"*, PhD thesis, Université Joseph Fourier, 2010. (Cité en page 7.)

[113] R. Reiter. "A logic for default reasoning", *Artificial Intelligence*, pages 81–132, 1980. (Cité en page 60.)

[114] R. Reiter. *"Knowledge in Action : Logical Foundations for Specifying and Implementing Dynamical Systems"*, MIT Press, 2001. (Cité en page 59.)

[115] J. Robert and R. Saffiotti. "Navigation by Stigmergy : A realization on an RFID Floor for minimalistic robots", *In. Proc of the IEEE Int Conf on Robotics and Automation (ICRA), Koba, Japan*, pages 245–252, 2009. (Cité en pages xiii et 11.)

[116] P. R. W. Roe, editor. *Towards an inclusive future - Impact and wider potential of information and communication technologies.* COST, Brussels, 2007. (Cité en page 22.)

[117] G.P Zarri, L. Sabri, A. Chibani, and Y. Amirat. "Semantic-Based Industrial Engineering : Problems and Solutions", In *Proceedings of the 4th Conference on Complex, Intelligent and Software Intensive Systems*, CISIS 2010, Cracovie, Poland, 2010. (Cité en page 142.)

[118] L. Sabri, A. Chibani, J. Beck, G.P Zarri, J.S. Brunner, Y. Amirat and P. Gatellier. "An Editor for Micro-Concept Rules Design", In *Proceedings of the 3rd International RuleML-2009 Challenge, collocated with the 3rd International Symposium on Rules, Applications and Interoperability*, RuleML 2009, Las Vegas, USA, 2009. (Cité en page 75.)

[119] L. Sabri, A. Chibani, Y. Amirat, and G.P Zarri. "Semantic and architectural approach for Spatio-Temporal Reasoning in Ambient Assisted Living", In *International Joint Conference on Artificial Intelligence*, IJCAI 2011, Barcelona, Spain, pages 77-84, 2011. (Cité en page 65.)

[120] L. Sabri, A. Chibani, Y. Amirat, and G.P Zarri. "Semantic-based situation recognition for ubiquitous robots and ambient intelligence environments", In *in : Journées Nationales de la Robotique Interactive du GDR Robotique*, JNRI 2011, Paris, France, 2011. (Cité en page 65.)

[121] A. Saffiotti, M. Broxvall, M. Gritti, K. Leblanc, R. Lundh, and J. Rashid. "the PEIS-Ecology project : Vision and results", In *In Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2008. (Cité en page 48.)

[122] E. Salvat and M.L. Mugnier. "Sound and Complete Forward and Backward Chainings of Graph Rules", In P. Eklund, G. Ellis, and G. Mann, editors, *Conceptual Structures : Knowledge Representation as Interlingua*, volume 1115, pages 248–262. Springer Berlin Heidlberg., 1996. (Cité en page 33.)

[123] Eric Salvat. *"Raisonner avec des opérations de graphes"*, PhD thesis, Université de Monpellier II, 1997. (Cité en page 33.)

[124] A. Sanfeliu, N. Hagita, and A. Saffiotti. "Network robot systems", *Robotics and Autonomous Systems*, volume 56, pages 793–79, 2008. (Cité en page 16.)

[125] K. Sang-Kyun, S. Mi-Young, K. Chul, Y. Sang-Jun, J. Hyun Chul, and L. Kyu-Chul. "Temporal ontology language for representing and reasoning interval-based temporal knowledge", In *Proceedings of the 3rd Asian Semantic Web Conference on The Semantic Web*, ASWC'08, pages 31–45, Berlin, Heidelberg, 2008. Springer-Verlag. (Cité en page 64.)

[126] S. Coradeschi and A. Saffiotti. "An introduction to the anchoring problem", In *Robotics and Autonomous Systems*, volume 43, pages 85-96, 2003. (Cité en page 49.)

[127] M. Daoutis, S. Coradeshi, and A. Loutfi. "Grounding commonsense knowledge in intelligent systems", In *Journal of Ambient Intelligence and Smart Environments*, pages 311-321, 2009. (Cité en pages 48 et 49.)

[128] J. Sayah. *"Contribution á la modélisation, á la simulation et á l'évaluation d'applications nomades á intelligence répartie - Application á l'assistance aux*

voyageurs dans les transports publics et les pôles d'échanges", PhD thesis, édoctorale mathématique et STIC, université PARIS-EST, 2009. (Cité en page 11.)

[129] M. Scheermesser. "Akzeptanz des Bewegungsmonitorings bei Chronischen Patienten", In *Proceedings of the Second German Congress on Ambient Assisted Living. VDE, Berlin, Germany*, 2009. (Cité en page 9.)

[130] B. Norman, A.N. Schilit, W. Roy . "Context-Aware Computing Applications", In *Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications*, pages 85–90, Washington, DC, USA, 1994. (Cité en page 19.)

[131] O. Scholz and T. Velten. "Integration von AAL-relevanter Sensorik in Zahntechnische Vorrichtungen", *In : Proceedings of the Second German Congress on Ambient Assisted Living. VDE, Berlin, Germany*, 2009. (Cité en page 9.)

[132] D. Schwabe, M. Heide, A. Neudeck, and U. Mehring. "aktive hüftprotektoren mit Telemonitoringfunktion" *In : Proceedings of the German Congress on Ambient Assisted Living. VDE, Berlin, Germany*, 2008. (Cité en page 9.)

[133] M. Shanahan. "The Event Calculus Explained" In M.J. Wooldridge and M. Veloso, editors, *Artificial intelligence today*, pages 409–430. Springer-Verlag, Berlin, Heidelberg, 1999. (Cité en page 60.)

[134] J. F. Sowa. "Conceptual graphs for a database interface," 1976. Available at `http://www.jfsowa.com/pubs/cg1976.pdf`. (Cité en page 29.)

[135] J. F. Sowa. "Conceptual structures : information processing in mind and machine", In *Addison-Wesley Longman Publishing Co., Inc.*, Boston, MA, USA, 1984. (Cité en page 33.)

[136] I. Horrocks. Available at `http://www.academia.edu/2686675/Ontologies_and_databases`. (Cité en page 64.)

[137] R. Stalnaker. "Presuppositions", volume 2, pages 77–96. Journal of Philosophical Logic, 1973. (Cité en page 108.)

[138] T. Strang and C. Linnhoff-Popien. "A context modeling survey", In *In : Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/England*, 2004. (Cité en page 41.)

[139] Rudi Studer, V. Richard Benjamins, and Dieter Fensel. "Knowledge Engineering : Principles and Methods", *Data Knowl. Eng.*, volume 25, pages 161–197, 1998. (Cité en page 27.)

[140] B. Suresh, A. Coates, and A.Y. Ng. "Autonomous sign reading for semantic mapping", In *Robotics and Automation (ICRA), 2011 IEEE International Conference*, pages 3297–3303, 2011. (Cité en page 48.)

[141] J.A. Tamada, M. Lesho, and M.J. Tierney. "Keeping watch on glucose :new monitors help fight the long-term complications of diabetes", *IEEE Spectr*, 39(4) :52–57, 2002. (Cité en page 9.)

[142] L. Sabri, A. Chibani, Y. Amirat, and G.P Zarri. "Narrative reasoning for cognitive ubiquitous robots", In *Knowledge Representation for Autonomous Robots, Workshop at IEEE/RSJ International Conference on Intelligent Robots and Systems*, IROS '11, San Fransisco, USA, 2011. (Cité en page 103.)

[143] J. Tappolet and A. Bernstein. "Applied temporal RDF : Efficient temporal querying of RDF data with SPARQL", *Proceedings of the 6th European Semantic Web Conference on The Semantic Web : Research and Applications*, pages 308–322, 2009. (Cité en page 61.)

[144] Y. Ren, J. Z. Pan, Y. Zhao. "Closed World Reasoning for OWL2 with NBox", In *T. Science and Technology*, volume 15, pages 692-701, 2010. (Cité en page 65.)

[145] M. Tenorth and M. Beetz. "KnowRob : knowledge processing for autonomous personal robots", IROS'09, pages 4261–4266, St. Louis, MO, USA, 2009. (Cité en pages xiii, 49, 52 et 54.)

[146] M. Uschold, M. Gruninger, M. Uschold, and M. Gruninger. *"Knowledge Engineering Review"*, volume 11, pages 93–136, 1996. (Cité en page 27.)

[147] J. Wahr and K. Tremper. "Non-invasive oxygen monitoring techniques," *Critical Care Clinics*, 11(1) :199–217, 1995. (Cité en page 9.)

[148] M. Waibel, M. Beetz, J. Civera, R. d'Andrea, J. Elfring, D. Galvez-Lopez, K. Haussermann, R.J.M. Janssen, J.M.M. Montiel, L. Perzylo, B. Schiessle, M. Tenorth, O. Zweigle, and M.J.G. Van de Molengraft. "RoboEarth - a world wide web for robots," volume 12, pages 69–82, 2011. (Cité en pages 49 et 54.)

[149] X.H. Wang, D.Q. Zhang, T. Gu, and H.K. Pung. "Ontology based context modeling and reasoning using OWL", *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, IEEE Computer Society*, 2004. (Cité en pages xiii et 44.)

[150] R. Want, A. Hopper, V. Falcão, and J. Gibbons. "The active badge location system," *ACM Trans. Inf. Syst*, volume 10, pages 91–102, 1992. (Cité en page 12.)

[151] C. Welty and R. Fikes. "A reusable ontology for fluents in owl," *Frontiers in Artificial Intellignece and Applications*, volume 150, pages 229–236, 2006. (Cité en pages 62 et 63.)

[152] W.A. Woods and J.G. Schmolze. "The Klone family," *Computers & Mathematics With Applications*, pages 133-177, 1992. (Cité en page 34.)

[153] F. Mastrogiovanni, A. Sgorbissa, R. Zaccaria. "A distributed architecture for symbolic data fusion," *In Prod. of the 20th Intl. Joint Conf. on Artificail Intelligence, AAAI*, pages 2153-2158, 2007. (Cité en page 47.)

[154] C. F. Crispim-Junior, V. Joumier, Y. L. Hsu , M. C. Pai, P. C. Chung, A. Dechamps, P. Robert, F. Bremond. "Alzheimer's patient activity assessment using different sensors", In *Gerontechnology*, pages 266-267, 2012. (Cité en page 8.)

[155] C. Matheus, M. Kokar, K. Baclawski, J. Letkowski, C. Call, M. Hinman, J. Salerno, D. Boulware. 'SAWA : An Assistant for Higher-Level Fusion and Situation Awareness", *SPIE Conference on Multisensor, Multisource Information Fusion, Orlando, Florida,*, 2005. (Cité en pages 44 et 47.)

[156] A. Yachir, Y. Karim, T.and Amirat, A. Chibani, and N. Badache. "qos based framework for ubiquitous robotic services composition", In *IROS*, pages 2019–2026, 2009. (Cité en page 48.)

[157] J. Ye, S. Dobson, and S. McKeever. "Situation Identification Techniques in Pervasive Computing : A review", *Pervasive and Mobile Computing*, volume 8, pages 36–66, 2012. (Cité en page 41.)

[158] H. Young-Guk, S. Joo-Chan, C. Young-Jo, and Y. Hyunsoo. "Towards ubiquitous robotic companion : Design and implementation of ubiquitous robotic service framework", In *ETRI Journal*, volume 27(6), pages 666–676, 2005. (Cité en page 48.)

[159] A. Zahneisen. "SOPHIA- best practice". *Proceedings of the Second German Congress on Ambient Assisted Living*, 2009. (Cité en page 9.)

[160] G.P. Zarri. "Representation and Management of Narrative Information : Theoretical Principles and Implementation (Advanced Information and Knowledge Processing)", Springer Verlag. series, 2009. (Cité en pages xiv et 106.)

[161] K. Zhou, K.M. Varadarajan, A. Richtsfeld, M. Zillich, and M. Vincze. "From holistic scene understanding to semantic visual perception : A vision system for mobile robots", In *Proc. of the Workshop at the 2011 IEEE International Conference on Robotics and Automation (ICRAW 2011)*, 2011. (Cité en page 48.)

[162] E. Bertino, R. Provetti, E. Salvetti. "Local Closed-World Assumptions for reasoning about Semantic Web data", In *Proceedings of the APPIA-GULP-PRODE Conference on Declarative Programming*, pages 314-323, 2003. (Cité en page 65.)

[163] J. Tao, S. Evren, J. Bao, and D.L. McGuinness. "Integrity Constraints in OWL", In *AAAI*, 2010. (Cité en page 64.)

[164] B. Motik, I. Horrocks, R. Rosati, U. Sattler. "Can OWL and logic programming live together happily ever after ?", In *the 5th international conference on The Semantic Web, ISWC'06* pages 501–514, Springer-Verlag 2006. (Non cité.)

[165] M. Kifer, G. Lausen. "F-logic : a higher-order language for reasoning about objects, inheritance, and scheme", In *Proceedings of the 1989 ACM SIGMOD international conference on Management of data, SIGMOD '89,* pages 134–146, 1989. (Cité en page 47.)

[166] J. Bohn, V. Coroama, M. Langheinrich, F. Mattern, and M. Rohs. "Living in a world of smart everyday objects", In *Social, Economic, and Ethical Implications*, volume 5, pages 763–786. 2004. (Cité en page 7.)

[167] Riboni, D. Bettini, C. M. Baldauf, S. Dustdar, and F. Rosenberg. "OWL 2 modeling and reasoning with complex human activities", In *Elsevier Science Publishers B. V*, volume 7, pages 1574-1192. In : Journal Pervasive Mob. Comput, 2011. (Cité en page 64.)

[168] T. Gu, H. K. Pung, D.Q. Zhang. "A service-oriented middleware for building context-aware services", In *Journal Netw. Comput. Appl.* , volume 7, pages 1084-8045. Academic Press Ltd, 2005. (Cité en pages xiii, 44 et 46.)

[169] D.A. Randell, Z. Cui, A.G. Cohn. " A spatial logic based on regions and connection", In *Proceedings of the 3rd International Conference on Knowledge Representation and Reasoning*, 1992. (Cité en page 47.)

[170] H.S. Il, H.L. Gi, H. Wonil, S. Hyowon, C. Jung-Hwa, P. Young-Tack. "Ontology-based multi-layered robot knowledge framework (OMRKF) for robot intelligence", In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems* , pages 429-436, 2007. (Cité en pages xiii et 55.)

[171] S. Coradeschi, A. Saffiotti. "Perceptual anchoring of symbols for action", In *Proceedings of the 17th international joint conferene on Artificial intelligence* , volume 1, pages 407–412. Morgan Kaufmann Publishers Inc, 2001. (Cité en page 49.)

[172] A. Bouguerra, K. Lars , A. Saffiotti. "Assessment for Sensor-Based Recovery Planning", In *Proc Situation of the 17th European Conf. on Artificial Intelligence (ECAI)* , 2006. (Cité en page 49.)

[173] S. Bragaglia, F. Chesani, P. Mello, D. Sottara. "A rule-based implementation of fuzzy tableau reasoning", In *Proceedings of the 2010 international conference on Semantic web rules (RuleML'10)* , 2010. (Cité en page 152.)

[174] N. Baumgartner, W. Gottesheim, S. Mitsch, W. Retschitzegger, W. Schwinger. "BeAware ! - Situation Awareness, the Ontology-Driven Way", In *Data & Knowledge Engineering* , volume 69, pages 1181-1193, 2010. (Cité en page 44.)

[175] H. Lausen, J. de Bruijn , A. Polleres, D. Fensel. "WSML - a Language Framework for Semantic Web Services", In *Position Paper for the W3C rules workshop, Washington DC, USA*, 2005. (Cité en page 65.)